UNIVERSITY OF CALIFORNIA,
IRVINE

Lossy Compression with Machine Learning: Techniques and Fundamental Limits

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Yibo Yang

Dissertation Committee:
Professor Stephan Mandt, Chair
Professor Erik Sudderth
Professor Weining Shen

2024

DEDICATION

To my family.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I want to thank my advisor Stephan Mandt for his excellent mentorship and continuous support. It was Stephan who introduced to me the fascinating topic of neural compression and generated many key ideas that set me on this research path. Under Stephan's mentorship, I was always given timely feedback and also plenty of intellectual freedom that allowed me to excel. Thank you for your trust and encouragement, for making me a better researcher and science communicator, for sharing your valuable insights and wisdom, and for being there with me every step of the way.

Similarly, I would like to thank Robert Bamler, who served as my secondary advisor at the beginning of my PhD. Although we did not get to spend much time together, you left a lasting impact on me with your meticulous style and dedication to science. Our time working together was one of the most productive years of my PhD and helped me grow tremendously as a researcher.

I would like to extend a special thanks to the neural compression team at Google, whose pioneering work influenced this thesis in many ways. Working with Lucas Theis was always a joy and learning experience, from collaboration on the neural compression monograph to insightful research discussions (one of which nudged me towards looking into ML approaches to information theory, as in Chapter 6). Jona Ballé laid much of the groundwork of this thesis (and for that matter, much of the current field of research) with their work on nonlinear transform coding, and continues to inspire me with their intellectual curiosity, creativity, and kindness. David Minnen was an excellent supervisor and mentor during my internship at Google Research, whose generous help made my first industry experience less overwhelming. David was also the one who got me really thinking about the computational obstacles facing neural compression, and our discussions eventually led me to work on shallow decoders in Chapter 5. I would also like to thank George Toderici, Nick Johnston, Sung Jin Hwang, Fabian Mentzer, and Eirikur Agustsson for all the help and inspiration.

This thesis would not be the same without many excellent collaborators. I had the fortunate and pleasure of working with Marcel Nutz and Stephan Eckstein, who taught me more than mathematics and contributed the theoretical foundation of my work in Chapter 7. I'm also grateful for my collaborators at UCI — Justus Will, Ruihan Yang, Jessica Howard, Daniel Whiteson, and outside of UCI — Yang Yang, and Joseph Marino.

I am very grateful for Drs. Alireza Mahkzani and Roger Grosse for expanding my research horizons and hosting me at the Vector Institute, and the many friends at the Vector Institute – Daniel Severo, Kirill Neklyudov, Rob Brekelmans, Buu Phan, Yangjun Ruan – for all the fun discussions and adventures. I'd also like to thank other friends in compression – Karen Ullrich, Ezgi Özyılkan, Berivan Isik, Greg Flamich, Lyndon Duong, and Daniel Severo (again) for their friendship and support. Thanks also go to the Mandt lab and other machine learning PhD students at UCI for their friendship and camaraderie, and faculty members at UCI, particularly Profs. Padhraic Smyth, Erik Sudderth, Weining Shen, Yaming Yu, and Alexander Ihler, for their mentorship and guidance during my PhD.

My journey in scientific research started nearly a decade ago with early forays into research during high school and college years. For that, I owe much to Sr. Cecilia Sehr of Bishop Lynch High School, Dr. Peng Tao of Southern Methodist University, my undergraduate research advisors Profs. Nicholas Ruozzi, John Hansen, Hynek Boril, Vibhav Gogate, Vincent Ng, as well as the Center for Language and Speech Processing at Johns Hopkins University. Thank you all for believing in my potential and giving me the research opportunities that ultimately led me to a PhD.

No words can describe my eternal gratitude for my family, 爸, 妈, Ava, Percy, Bill, Lauren, and my extended family. Thank you for your unconditional love and support, for all my fondest memories, for shaping me into who I am, and for giving me a home that I can always return to wherever I end up in this world. A special thanks to Antonios, Andreja, and the swing dance club at UCI for enriching my life outside of the PhD. Lastly, I want to thank the Vargas, Mr. Kirsch, Mr. Morrow, and many, many more people who helped me along the way.

The work in thesis was supported by a fellowship from the Hasso Plattner Foundation and funding from the National Science Foundation (NSF) under the NSF CAREER Award IIS2047418 and IIS-2007719.

VITA

Yibo Yang

EDUCATION

Master of Science in Computer Science — **2021**
University of California, Irvine — *Irvine, California*

Bachelor of Science in Computer Science — **2018**
University of Texas at Dallas — *Dallas, Texas*

RESEARCH EXPERIENCE

Graduate Research Assistant — **2019 – 2024**
University of California, Irvine — *Irvine, California*

Research Intern — **2024**
Amazon Web Services — *New York, New York*

Research Intern — **2023**
The Vector Institute for AI — *Toronto, Canada*

Research Intern — **2022**
Google Research — *Mountain View, California*

Research Assistant — **2017 – 2019**
University of Texas at Dallas — *Dallas, Texas*

Research Intern — **2016**
Johns Hopkins University — *Baltimore, Maryland*

TEACHING EXPERIENCE

Teaching Assistant — **2023**
University of California, Irvine — *Irvine, California*

Teaching Assistant — **2018**
University of Texas at Dallas — *Dallas, Texas*

REFEREED JOURNAL PUBLICATIONS

Learning to Simulate High Energy Particle Collisions From Unlabeled Data **2022**
Jessica N Howard, Stephan Mandt, Daniel Whiteson, and Yibo Yang. *Scientific Reports*

Insights from Generative Modeling for Neural Video Compression **2023**
Ruihan Yang, Yibo Yang, Joseph Marino, and Stephan Mandt. *Transactions on Pattern Analysis and Machine Intelligence*

An Introduction to Neural Data Compression **2023**
Yibo Yang, Stephan Mandt, Lucas Theis. *Foundations and Trends in Computer Graphics and Vision*

REFEREED CONFERENCE PUBLICATIONS

Variational Bayesian Quantization **2020**
Yibo Yang, Robert Bamler, and Stephan Mandt. *International Conference on Machine Learning (ICML)*

Improving Inference for Neural Image Compression **2020**
Yibo Yang, Robert Bamler, and Stephan Mandt. *Conference on Neural Information Processing Systems (NeurIPS)*

Hierarchical Autoregressive Modeling for Neural Video Compression **2021**
Ruihan Yang, Yibo Yang, Joseph Marino, Stephan Mandt. *International Conference on Learning Representations (ICLR)*

Towards Empirical Sandwich Bounds on the Rate-Distortion Function **2022**
Yibo Yang, and Stephan Mandt. *International Conference on Learning Representations (ICLR)*

Computationally-Efficient Neural Image Compression with Shallow Decoders **2023**
Yibo Yang, and Stephan Mandt. *International Conference on Computer Vision (ICCV)*

Estimating the Rate-Distortion Function by Wasserstein Gradient Descent **2023**
Yibo Yang, Stephan Eckstein, Marcel Nutz, Stephan Mandt. *Conference on Neural Information Processing Systems (NeurIPS)*

Progressive Compression with Universally Quantized Diffusion Models **2025**
Yibo Yang, Justus Will, and Stephan Mandt. *International Conference on Learning Representations (ICLR)*

CONFERENCE PUBLICATIONS PRIOR TO PHD STUDIES

One-Shot Marginal MAP Inference in Markov Random Fields **2019**
Hao Xiong*, Yuanzhen Guo*, Yibo Yang*, and Nicholas Ruozzi. *Uncertainty in Artificial Intelligence (UAI)*

Lifted Hybrid Variational Inference **2020**
Yuqiao Chen*, Yibo Yang*, Sriraam Natarajan, and Nicholas Ruozzi. *International Joint Conference on Artificial Intelligence (IJCAI)*

* denotes equal contribution.

ABSTRACT OF THE DISSERTATION

Lossy Compression with Machine Learning:
Techniques and Fundamental Limits

By

Yibo Yang

Doctor of Philosophy in Computer Science

University of California, Irvine, 2024

Professor Stephan Mandt, Chair

The exponential growth in global data creation and transmission necessitates increasingly powerful data compression algorithms. *Neural* compression, which leverages neural networks and deep learning techniques, presents a promising new frontier by learning compression algorithms end-to-end from data. This approach has shown great potential across various data types, outperforming traditional codecs that have been hand-designed over decades. Despite its promise, neural compression still faces many open problems and roadblocks. This thesis focuses on neural *lossy* compression, and tackles the problems of inference sub-optimality, high computation complexity, and establishing fundamental bounds for performance evaluation. Conceptually our contributions are two-fold, concerning both the practical performance and theoretical performance limits of neural lossy compression:

- First, drawing on techniques from deep learning and probabilistic inference, we develop methods for improving the practical performance of neural lossy compression, pushing its boundaries in rate-distortion performance and computation efficiency. We identify approximation gaps in the existing nonlinear transform coding [Ballé et al., 2021] approach, and propose inference optimization algorithms that close these gaps using ideas from variational optimization, the Gumbel-softmax trick, and bits-back coding. The proposed Stochastic Gumbel Annealing (SGA), allows for plug-and-play optimization of the discrete latent representation at inference time, and is shown to significantly improve the rate-distortion performance of the base approach. The idea of improved inference (or encoding) also lends itself to a novel asymmetric autoencoder architecture, which shifts much of the decoding computation cost to a higher encoding cost up-front. We design a lightweight and shallow decoder inspired by traditional transform coding algorithms like JPEG, and pair it with a powerful encoding procedure (such as SGA), demonstrating rate-distortion performance competitive with the Mean-Scale Hyperprior architecture [Minnen et al., 2018] at a fraction of the decoding computation cost.

- Second, building on the interplay between lossy compression, statistical estimation, and

entropic optimal transport, we present computational methods for estimating the rate-distortion (R-D) function, a fundamental quantity in the theory of lossy compression which delineates the performance ceiling of lossy compression for a given data source. We develop novel variational bounds which sandwich the R-D function and are amenable to deep learning tools, as well as a neural-network-free, particle-based upper bound of the R-D function using optimal transport (OT) ideas. These algorithms allow for the estimation of the R-D function on a much larger class of problems than previously possible with the classic Blahut-Arimoto [Blahut, 1972, Arimoto, 1972] algorithm, such as on continuous real-world data sources. We empirically obtain tight sandwich bounds on the R-D functions of real-world sources from particle physics, speech, and GAN-generated images. We also obtain an R-D upper bound for high-resolution natural images, and shed light on the optimality of recent lossy image compression algorithms. On the theoretical side, we characterize the convergence and sample complexity of our particle-based method, and offer new insight into the solution of the R-D problem in the continuous setting based on connections to entropic optimal transport and maximum-likelihood deconvolution.

While lossless compression and machine learning have long been understood as two sides of the same coin — any probability model of discrete data gives rise to a lossless compressor of the data source and vice versa [Kraft, 1949, McMillan, 1956], and compression-related metrics such as cross-entropy and bits-per-dimension routinely appear in machine learning [Bishop, 2006, Theis et al., 2016] — less known is the connection between *lossy* compression and machine learning. This thesis explores and further develops the parallel between lossy compression and machine learning, where a rate-distortion loss is equivalently viewed as a variational inference/learning objective; this objective can be naturally motivated by the statistical problem of maximum-likelihood deconvolution, and is further related to the cost of entropic-OT projection. It is our goal and hope for these connections to inspire and guide the

development of new learning-based compression algorithms, as well as facilitate the exchange of ideas between the fields of compression, information theory, and machine learning.

Chapter 1

Introduction

The storage and transmission of information is fundamental to nearly every aspect of today's digital world, from capturing and streaming high-definition videos to enabling the operations of financial markets and autonomous vehicles. With the exponential growth in global data creation and transmission [BusinessWire, 2021] and the emergence of new media applications such as Virtual and Augmented Reality, efficient data compression has never been more critical.

At its core, compression works by exploiting the structure and patterns inherent in the data, so that more predictable patterns can be described with fewer bits. It is then no surprise that statistical learning, which aims to understand structure in data, is built into such celebrated compression algorithms as adaptive arithmetic coding [Rissanen and Langdon, 1979, Witten et al., 1987] and Lempel-Ziv [Ziv and Lempel, 1977, 1978] (the basis of extremely popular file formats such as `zip` and `gzip`). In the reverse direction, compression ideas have also profoundly influenced machine learning, starting from Occam's razor, to the principles of Minimum Description Length [Grünwald, 2007] and Information Bottleneck [Tishby et al., 2000, Tishby and Zaslavsky, 2015], and to the current Large Language Models whose roots

trace back to Shannon's landmark treatise on information theory [Shannon, 1948].

In recent years, the development of sophisticated statistical models (i.e., "deep generative models") such as VAEs [Kingma and Welling, 2014], GANs [Goodfellow et al., 2014], and normalizing flows [Papamakarios et al., 2021] is having an increasing impact on data compression. Information theory tells us that any probability model for discrete data corresponds to a lossless compression algorithm [Kraft, 1949, McMillan, 1956], and various approaches have been developed for neural lossless compression based on (variational) autoencoders [Mentzer et al., 2019, Townsend et al., 2019a], normalizing flows [Hoogeboom et al., 2019, Tran et al., 2019, Ho et al., 2019a], autoregressive models [Hoogeboom et al., 2021a], and diffusion models [Kingma et al., 2021]. Ideas for neural lossy compression can already found in early work on hierarchical latent variable models [Gregor et al., 2016], and are subsequently made competitive in rate-distortion performance by end-to-end training with an effective entropy model and approximation to quantization [Ballé et al., 2016, Theis et al., 2017a]. Subsequent methods further incorporate GAN [Agustsson et al., 2019, Mentzer et al., 2020] or diffusion [Theis et al., 2022, Ghouse et al., 2023, Hoogeboom et al., 2023, Yang and Mandt, 2023a, Yang et al., 2025] for enhanced realism or perceptual quality. Neural lossy codecs have gradually surpassed traditional codecs in compression performance across diverse data types, ranging from images [Ballé et al., 2017, Minnen and Singh, 2020, Yang et al., 2020a, He et al., 2022], audio [Zeghidour et al., 2021, Défossez et al., 2022], video [Lu et al., 2019, Agustsson et al., 2020, Yang et al., 2021, Mentzer et al., 2022], to point clouds [Guarda et al., 2019, Quach et al., 2022] and 3D scenes [Tang et al., 2020, Bird et al., 2021].

Despite its potential, neural compression still faces many open problems. Chief among them is its high computation complexity compared to traditional codecs, limiting its deployment in real-world environments with strict latency or resource constraints such as mobile devices or streaming services [Minnen, 2021, Mukherjee, 2022]. The solution requires simultaneously improving the compression performance and minimizing the computation

complexity, potentially entailing a trade-off between the two. Classical information theory does not account for computational constraints, and currently there is no general theory that can guide the practical development of low-complexity codecs. It is also unclear how well today's compression methods perform relative to the information-theoretic performance limit of compression. For example, in the case of lossy compression, if we are in fact already approaching the performance limit given by Shannon's rate-distortion function [Shannon, 1959] or its refined versions [Kontoyiannis, 2000, Kostina, 2016], we should focus our attention on optimizing other perhaps less-researched performance criteria such as realism [Blau and Michaeli, 2018, Blau and Michaeli, 2019] or computation complexity [Johnston et al., 2019, Yang and Mandt, 2023b], or the development of better criteria themselves [Qiu et al., 2024]. However, the estimation of fundamental information-theoretic quantities such as entropy and the rate-distortion function requires solving difficult mathematical optimization problems, for which traditional algorithms such as the Blahut-Arimoto algorithm [Blahut, 1972, Arimoto, 1972] are ill-equipped. The work in this thesis makes advances on these problems.

Overview and Contributions

This thesis presents learning-based techniques for improving lossy compression, both in terms of practical performance and theoretical understanding. The thesis is structured as follows.

We begin with a self-contained introduction to information theory and neural compression. Chapter 2 reviews the fundamentals of data compression, i.e., source coding, covering core topics such as entropy coding, rate-distortion theory, and transform coding. This part can be skipped by readers with a background in information theory. Chapter 3 introduces basic approaches to *neural* data compression, starting with lossless compression (e.g, neural entropy models) and culminating in nonlinear transform coding. Here, we highlight the connection between variational inference/learning and (neural) compression: in the lossless case, the

"bridge" is given by bits-back coding; in the lossy case, the "bridge" is given by universal quantization for a sub-family of latent variable models (those that implement nonlinear transform coding), and is given by relative entropy coding for more general latent variable models. After the background Chapters 2 and Chapter 3, the main contributions of the thesis can be organized conceptually in two main parts, consisting of two chapters each:

Part I ("practical techniques", or "machine learning for data compression") consists of Chapters 4 and 5, and develops techniques for improving the practical performance of neural lossy compression, in terms of rate, distortion, and decoding complexity. Specifically, Chapter 4 takes the variational inference view of nonlinear transform coding, and identifies and addresses three common approximation gaps which hinder lossy compression performance: an amortization gap, a discretization gap, and a marginalization gap. The amortization gap and discretization gap arise from the inability of the encoder network to predict the optimal discrete latent representation from each given input data instance, which entails a discrete optimization problem in high dimensions. We propose to close these two gaps with a novel algorithm called Stochastic Gumbel Annealing (SGA) based on iterative inference and a variational approximation to quantization, which can navigate the discrete optimization problem much more effectively than alternative methods. The marginalization gap stems from using a two-part code [Grünwald, 2007] to losslessly compress the discrete latent representation in a hierarchical model, and is closed by doing bits-back coding instead. The proposed inference optimization algorithms are shown to significantly improve the rate-distortion performance of image compression with the Mean-Scale Hyperprior architecture [Minnen et al., 2018], by $15\% \sim 20\%$ on standard benchmarks.

Chapter 5 extends the rate-distortion criteria of neural lossy compression to also optimize for decoding computation complexity, a key requirement for practical applications such as loading images on a mobile phone. Inspired by low-complexity traditional transform coding algorithms such as JPEG, we replace the deep convolutional synthesis transform of nonlinear

transform coding with extremely shallow and lightweight transforms. Building on the idea of inference optimization in Chapter 4, we theoretically show that a powerful encoding procedure can compensate for a cheaper decoder in rate-distortion performance by reducing an inference gap. We empirically show that the resulting asymmetric autoencoder architecture, optionally combined with an iterative encoding procedure such as SGA, drastically expands the rate-distortion-complexity frontier of image compression with nonlinear transform coding, preserving the rate-distortion performance of the popular Mean-Scale Hyperprior [Minnen et al., 2018] model with a five-fold reduction in decoding complexity.

Part II ("fundamental limits", or "machine learning for information theory") consists of Chapters 6 and 7, and presents computational methods for estimating the fundamental performance limit of lossy compression delineated by Shannon's rate-distortion function [Shannon, 1959], $R(D)$. Specifically, Chapter 6 develops the first sandwich bounds on the rate-distortion function that scale to continuous and high-dimensional sources, for which the classic Blahut-Arimoto algorithm [Blahut, 1972, Arimoto, 1972] is infeasible. The overall idea is to formulate the variational problem defining $R(D)$ in terms of one that can be solved with stochastic optimization over a class of functions parameterized by neural networks. The proposed variational upper bound is directly related to β-VAEs [Higgins et al., 2017] with a particular likelihood model, and precisely establishes such latent variable models as estimators of the source $R(D)$ function. The proposed lower bound is inspired by a dual formulation of $R(D)$ from Csiszár [1974a], and represents the more challenging direction of the sandwich bound that requires significantly more effort to estimate. Experimentally, we estimate $R(D)$ sandwich bounds that appear tight for various sources with low effective dimension (but possibly high ambient dimension), as well as $R(D)$ upper bounds on high-resolution natural images, suggesting considerable room for improvement in the rate-distortion performance of neural image compression methods.

Finally, Chapter 7 introduces optimal transport ideas and insights into the $R(D)$ estimation

problem, which complements the deep learning approach from the earlier Chapter 6. We focus on the continuous case (i.e., the source and reproduction alphabets are $\mathbb{R}^d$), and introduce a neural-network free $R(D)$ upper bound estimator based on simulating the gradient flow [Ambrosio et al., 2008] of the *rate functional* [Harrison and Kontoyiannis, 2008]. Unlike the classic Blahut–Arimoto algorithm [Blahut, 1972, Arimoto, 1972] which fixes the support of the reproduction distribution in advance, our Wasserstein gradient descent algorithm learns the support of the optimal reproduction distribution by moving particles. Empirically, our particle-based approach requires minimal tuning compared to neural-network-based approaches, and offers significantly faster convergence when the latter struggle. This allows us to obtain tighter $R(D)$ upper bounds than the VAE approach in the previous chapter without designing data-specific encoder/decoder neural network architectures. Theoretically, we characterize the convergence of Wasserstein gradient descent, and offer new insight into the solution of the R-D problem based on connections to entropic optimal transport and maximum-likelihood deconvolution. Specifically, we show that the R-D problem is equivalent to that of minimizing an entropic optimal transport (EOT) cost between a variational reproduction measure and the source measure, allowing us to turn statistical bounds for EOT [Mena and Niles-Weed, 2019] into finite-sample bounds for R-D estimation. We also demonstrate the connection between the R-D problem and maximum likelihood deconvolution, showing that a Gaussian convolution of *any* distribution can serve as a source with a known solution to the R-D problem. We thus introduce a new, rich class of sources with known ground truth, including Gaussian mixtures, as a benchmark for algorithms.

The common background required for the thesis is non-linear transform coding (Section 3.2.1), which mainly builds on Chapter 2 and the discretized density model (Section 3.1.1). Chapter 4 additionally builds on bits-back coding (Section 3.1.3), and Chapter 6 also references the basic connection between latent variable modeling and bits-back coding (Section 3.1.3). Chapters 5, 6, and 7 make brief references to the idea of relative entropy coding (Section 3.2.2).

A word on notation

Given that this thesis spans multiple topics (machine learning, data compression, information theory, and even elements of optimal transport), each with its own notational convention that often conflicts with the others, it is difficult to come up with a unified notation that would not appear foreign or contrived to at least some readers. Therefore we let the context and the topic of each chapter dictate the notation convention, and make every effort to ensure notation consistency within each chapter (but not necessarily across different chapters).

There are mainly three flavors of notation in this thesis. The first notation is common in machine learning and applied work, and is used in Chapters 3, 4 and 5. The convention is to use bold letters to emphasize vectors ($\mathbf{x}$, $\mathbf{z}$, etc.) as opposed to scalars (x_i, z_j, etc.), use the same lower case letter ($\mathbf{x}$) often for both a random object ($\mathbf{X}$) and its realization ($\mathbf{x}$), and use lower-case p (or q) to denote either a probability mass or density function (and by extension the probability distribution/measure). The existing notation in neural compression literature largely follows this notation, except some differences during the discussion of nonlinear transform coding which borrows notation from signal processing [Gersho and Gray, 2012]. In nonlinear transform coding, the letters $\mathbf{y}$ and $\mathbf{z}$ often denote the output of an encoding (analysis) transform without a probabilistic interpretation, whereas in probabilistic machine learning, the letter $\mathbf{z}$ (and sometimes $\mathbf{y}$) conventionally refers to latent variables (e.g., [Kingma and Welling, 2014], [Beal and Ghahramani, 2003a]).

The second flavor of notation is more common in information theory and largely follows the convention in the textbook of [Polyanskiy and Wu, 2022]; it is used in Chapters 2, 6, and 7. Here the data source is a random object X taking value x in some abstract space $\mathcal{X}$; x can be a scalar, vector, or belong in some more exotic set, but the theory is the same as long as the source has a presumed probabilistic structure (most commonly *stationary* and *memoryless*, or *i.i.d.*). The probability measure induced by any random object X (which is

the pushforward of a base probability law $\mathbb{P}$ under X) is denoted P_X, which also double-duties as the probability mass function in the discrete case. The notation $P_{Y|X}$ denotes a (Markov) transition kernel, which be thought of as defining a conditional distribution $P_{Y|X=x}$ for every $x \in \mathcal{X}$.

The third flavor of notation is from optimal transport and only limited to Chapter 7. There the emphasis is on probability measures (μ, ν, etc.) and transition kernels (K), and less so on random variables which define the measures at the beginning and are forgotten afterwards. The notation will be explained as it appears in Chapter 7.

When special attention is deserved, we use a box like this:

> A word on notation: ...

Chapter 2

Background: Source Coding Fundamentals

We start by introducing basic background in data compression. Most of the material is classical and can be found in information theory textbooks like [Cover and Thomas, 2006] or [Polyanskiy and Wu, 2022]. A significant portion of this material has been published in [Yang et al., 2023a].

Depending on whether the compressed data can be recovered perfectly, compression can be categorized as either *lossless* or *lossy*. Lossless compression generally involves building a probability model of the data, based on which we can assign shorter codewords to more frequently observed outcomes. Lossless compression is often used for medical records, spreadsheets, and file archives such as `zip`. By contrast, lossy compression produces imperfect reconstructions and discards some information, e.g., via a quantization step. This entails a fundamental tradeoff between the bit-rate and fidelity of the data reconstruction. Lossy compression is widely used in the compression of digital media such as audio, images, and video.

2.1 The Overall Problem of Communication

To give a broader perspective, data compression was originally studied as part of a general theory of communication [Shannon, 1948], which includes both source coding (i.e., compression) and channel coding. As Shannon succinctly put it, "the fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point" [Shannon, 1948]. In the communication problem, an information *source* aims to communicate a message (or, data) to a destination over a physical medium known as a *channel*. Figure 2.1 illustrates the communication system proposed by Shannon for solving this problem, and is still the basis of communication systems today.

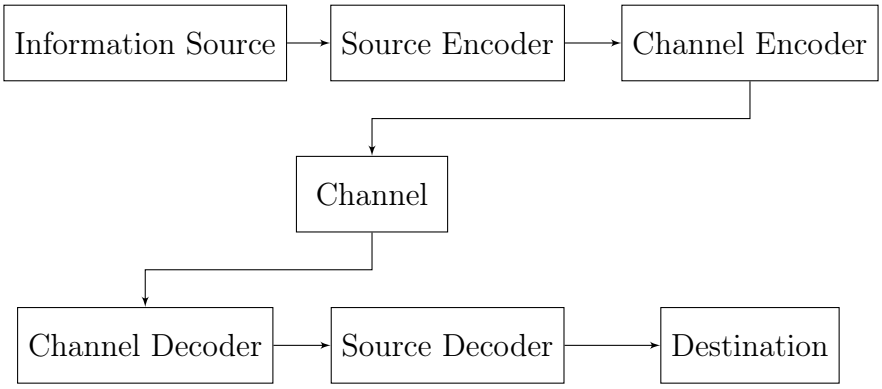


Figure 2.1: A conceptual diagram of the communication system conceptualized by Claude Shannon.

The *information source* produces a message, which is a sequence of values in some predetermined set $\mathcal{X}$. The *channel* is a physical medium which maps a given input value to an output value, and transmits information in the process. The *source encoder* takes a source message and converts it into a sequence of values that can be carried over the channel. Channels for digital transmission are typically binary, with input and output values in $\{0, 1\}$, whereas an analogue channel generally converts a real-valued input to a real-valued output. A basic challenge of communication is that channels are often imperfect or *noisy*, so that the output of the channel may not be identical to the input. E.g., bits stored on a magnetic hard disk might get flipped or corrupted over time, and a radial signal traveling in space can

be subject to interference from other signals or atmospheric conditions. To ensure reliable communication, the *channel encoder* therefore adds redundancy to the output of the source encoder in some way, so that the *channel decoder* can correctly recover the channel input with a high probability. Finally, the channel decoder's output is fed to the *source decoder*, which generates a reconstruction of the original source message intended for the destination.

The design of channel encoding and decoding schemes belongs to the study of *channel coding*, a centerpiece of information and communication theory. In this thesis we will not be concerned with channel coding, and focus instead on *source coding*. We will assume a perfectly noiseless binary channel, and try to come up with the best source encoder and decoder. The general goal of channel coding is to maximize the bit-rate of reliable communication across the noisy channel, whereas the goal of source coding is to minimize the bit-rate at which information is produced and to be transmitted in the first place.[1] The separation of channel and source coding can be theoretically shown to be asymptotically optimal [Shannon, 1948], and greatly simplifies the design and engineering of communication systems. The binary channel is used universally today for representing and transmitting all kinds of data, ranging from bank account details across the ATM network to multi-media gaming content over the Internet.

We treat the (typically unknown) *source* as (i.i.d. copies of) a random object X, which takes values in a predetermined set $\mathcal{X}$ also called the *alphabet*. A sequence of realizations of the source $(x_1, x_2, ..., x_n) \in \mathcal{X}^n$ is called a *message*. More generally, the source can be any discrete-time stochastic process $(X_1, X_2, ...,)$, rather than i.i.d. copies of the same X (i.e., a stationary and memoryless process).

Shannon's key innovation is the rigorous application of probability theory to a mathematical study of information. However, translating real-world phenomena into mathematical models comes with some difficulties. Since we rarely have knowledge of the true data generating

[1]The first use of the word "bit-rate" in this sentence refers to bits per channel use, while the second use "bit-rate" refers to bits per source symbol produced by the *source encoder* or *compressor*.

process, the assumptions we make about the source are a non-trivial design choice, and can have a sizable impact on compression performance.[2] Consider an example from [Sayood, 2017, Chapter 2]: given the following (rather contrived) sequence,

$$1, 2, 1, 2, 3, 3, 3, 3, 1, 2, 3, 3, 3, 3, 1, 2, 3, 3, 1, 2$$

what is the best way to compress it? When considered one digit at a time, the sequence seems to contain no obvious structure; assuming an i.i.d. scalar source with the alphabet $\{1, 2, 3\}$, the entropy of the source (i.e., "average compression cost") works out to be 1.5 bits per symbol. But if we assume an i.i.d. vector source taking values in the alternative alphabet $\{(1, 2), (3, 3)\}$, the entropy is reduced to 1 bit per symbol. [3] One can even argue that there is nothing inherently random about this given and fixed string, and the "compressibility" or "complexity" of such a string should not depend on a presumed probability model at all. Following this logic further would lead us to *algorithmic information theory* [Grünwald et al., 2008], which has intimate connections to the Minimum Description Length [Rissanen, 1978] approach to learning and model selection.

We will not dwell on this subtlety much. In this thesis, we mostly follow the convention in machine learning and assume an i.i.d. vector-valued source X has been given to us. We make no assumptions about the statistical relations among the scalar coordinates of X, and our assumption of i.i.d. vectors is usually benign especially if X is of high enough dimension. For example, X may correspond to an MNIST image, which itself is a two-dimensional stochastic process; but a sequence of images randomly drawn from the MNIST dataset $X_1, X_2, ...$ are reasonably considered independent.

The information-theoretic treatment of a non-i.i.d. source (commonly, a stationary and ergodic

[2]The equivalent problem in machine learning and statistics is that of *model selection*, which has a direct impact on generalization performance.

[3]This question has been recently re-examined in the context of tokenization for large language models [Rajaraman et al., 2024, Phan et al., 2024].

process) is outside the scope of this introduction, and can be found in e.g., [Polyanskiy and Wu, 2022, Chapters 6, 12].

2.2 Lossless Compression

Lossless compression aims to represent discrete data with as few bits as possible, while being able to perfectly recover the data later. More precisely, to each possible message $(x_1, x_2, ..., x_n)$ produced by the source X, our goal is to assign a unique string of bits (which we call a *bit-string*) so as to minimize the average number of bits required. Such an assignment is known as a lossless *source code*. Since the set of all possible bit-strings $\{0,1\}^* := \{\emptyset, 0, 1, 00, 01, ...\}$ is countable and we require a unique bit-string for every possible message, lossless compression is only possible for a discrete source X. The structure of $\mathcal{X}$ is not important for our conceptual discussions, and we can, without loss of generality, identify $\mathcal{X}$ with the set of natural numbers. Still, a reduction in the number of bits is never guaranteed, and lossless compression relies on the fact that certain outcomes of the source are more probable than others and can be assigned shorter bit-strings. Therefore, we cannot expect to compress a "pure noise" source whose outcomes are all equiprobable. Fortunately, real-world sources (such as the English language) often contain structures that makes a small subset of outcomes much more likely than others (e.g., the string "Hello world!" v.s. "bwzjnhxiHcBuq").

To quantify the amount of uncertainty in a source, Shannon made the following definition: for a discrete random variable X with probability mass function (PMF) P_X, its (Shannon) entropy is

$$H[X] := \mathbb{E}\left[\log_2 \frac{1}{P_X(X)}\right] \tag{2.1}$$

$$= \sum_{x \in \mathcal{X}} P_X(x) \log_2 \frac{1}{P_X(x)}. \tag{2.2}$$

For each outcome $x \in \mathcal{X}$, the number $\log_2 \frac{1}{P_X(x)}$ is known as *self-information* and quantifies the amount of surprise of observing $X = x$; entropy is therefore the expected value of surprise. For example, the entropy of a fair coin flip — a Bernoulli(0.5) random variable — is 1 bit, and the entropy of a biased coin flip approaches 0 bits as the probability of one of the two outcomes approaches 1. The definition of entropy can be motivated by natural axioms of "information content" [Shannon, 1948] and turns out to delineate the fundamental performance limit of lossless compression. In the rest of this section, we will give a mathematical description of this performance limit and introduce algorithms that aim to approach this limit, known as *entropy coding* algorithms.

2.2.1 Optimal Symbol Code

We start by considering assigning a bit-string to one symbol of the message at a time, i.e., we want to find a *symbol code* $\gamma : \mathcal{X} \to \{0,1\}^*$, and obtain the bit-string for an entire message by concatenation. We call each assigned bit-string $\gamma(x)$ for $x \in \mathcal{X}$ a *codeword*; we call the number of bits making up $\gamma(x)$ the *code length*, variously denoted by $l(x)$ or $|\gamma(x)|$. For the assignment to be lossless, clearly γ needs to assign a unique codeword to each x, so that the source decoder can recover x unambiguously from $\gamma^{-1}(\cdot)$. To allow the efficient decoding of a message from a sequence of concatenated codewords, we impose the further condition that γ is *prefix-free* [4], i.e., no codeword under γ is a prefix of another.

[4]Such code is also commonly referred to as a "prefix code", or "instantaneous code".

It turns out that any prefix code with code lengths l must satisfy the following *Kraft inequality*,

$$\sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1,$$

and conversely, a prefix code can always be constructed from code lengths that satisfy the above inequality [Cover and Thomas, 2006, Theorem 5.2.1].[5] As a consequence, the optimal average code length among all prefix (or uniquely decodable) codes is given by the following optimization problem:

$$\begin{aligned} L^*(X) := \min_{\gamma} & \sum_{x \in \mathcal{X}} P_X(x) l(x), \\ \text{s.t.} & \sum_{x \in \mathcal{X}} 2^{-l(x)} \leq 1, \\ & l(x) \in \mathbb{N}. \end{aligned}$$

By studying this problem, we arrive at the following basic result concerning the optimal average code length of prefix codes [Cover and Thomas, 2006, Theorem 5.4.1]:

$$H[X] \leq L^*(X) \leq H[X] + 1. \tag{2.3}$$

I.e., no prefix code (or more generally, no *uniquely decodable* code) can achieve an average code length below the source entropy, and there exists a prefix code whose average code length is one bit within entropy. This is an example of a *coding theorem*, which relates the mathematical definition of entropy to the operational cost of optimal lossless compression.

[5] The result also holds if we replace "prefix code" with "uniquely decodable code" (see [Cover and Thomas, 2006, Chapter 5.5]). Although the former is a subset of the latter, it turns out we do not lose any coding efficiency by considering prefix codes only.

2.2.2 Lossless Coding and Estimation

In practice, the true distribution P_X of the source is almost always unknown. We therefore approximate it by an estimated version P_θ, and proceed with entropy coding as if the data came from the distribution P_θ. Using a code designed for P_θ with code length $l(x)$, the average code length of representing $X \sim P_X$ with the wrong code then satisfies [Cover and Thomas, 2006, Theorem 5.4.3]

$$\mathbb{E}_{x\sim P_X}[l(x)] \geq H[X] + D_{\mathrm{KL}}(P_X \parallel P_\theta), \tag{2.4}$$

$$\mathbb{E}_{x\sim P_X}[l(x)] \leq H[X] + D_{\mathrm{KL}}(P_X \parallel P_\theta) + 1, \tag{2.5}$$

where D_{KL} denotes the *relative entropy*, or Kullback-Leibler (KL) divergence, defined in the discrete case as

$$D_{\mathrm{KL}}(P \parallel Q) = \mathbb{E}_{x\sim P}[-\log_2 Q(x) + \log_2 P(x)].$$

The KL divergence is always positive unless $P = Q$ and serves as an asymmetric distance measure between the two distributions.

Ignoring the (up to 1 bit) overhead of entropy coding above, the average coding cost of data from P_X under an imperfect model P_θ is therefore given by the *cross entropy*:

$$H[P_X, P_\theta] := \mathbb{E}_{x\sim P_X}[\log_2 \frac{1}{P_\theta(x)}],$$

which relates to relative-entropy via

$$H[P_X, P_\theta] = H[X] + D_{\mathrm{KL}}(P_X \parallel P_\theta).$$

This tells us that cross entropy is a variational upper bound on the ideal compression cost

$H[X]$, and the looseness in the bound — the relative entropy $D_{\mathrm{KL}}(P_X \parallel P_\theta)$ — captures the average "excess" code length compared to the ideal $H[X]$. Fitting a model P_θ on data from P_X using maximum-likelihood estimation, i.e.,

$$\arg\max_{P_\theta} \mathbb{E}_{x\sim P_X}[\log_2 P_\theta(x)] = \arg\min_{P_\theta} \mathbb{E}_{x\sim P_X}[-\log_2 P_\theta(x)], \tag{2.6}$$

is therefore precisely aligned with minimizing the coding cost of data from P_X under an imperfect model P_θ.

A caveat: discussions of entropy coding algorithms often center around the ideal case where the true P_X is known and given; in practice however, a probability model P_θ is often estimated first, and serves as a drop-in replacement for the unknown P_X when assigning codewords. The resulting average code length will therefore always suffer an overhead of at least $D_{\mathrm{KL}}(P_X \parallel P_\theta)$.

2.2.3 Huffman Coding

As an example, we review one of the most widely used entropy coding algorithms due to Huffman [1952], which achieves the optimal prefix code length in (2.3). In a nutshell, given a probability mass function (PMF) P_X over symbols, Huffman coding first builds a so-called Huffman tree (Figure 2.2), and then assigns to every symbol a unique binary codeword according to the tree.

The leaf nodes of the tree are associated with the symbols in the alphabet. A Huffman tree can be recursively grown from the leaves to the root by successively merging the two nodes with the lowest probabilities. By traversing the tree from the root to a leaf, the sequence of branching directions ("0" for left, "1" for right, in our example) assigns a unique binary codeword to each symbol, with the more frequent symbols receiving shorter codewords. Encoding and decoding are simple lookup operations with complexity linear in the tree depth.

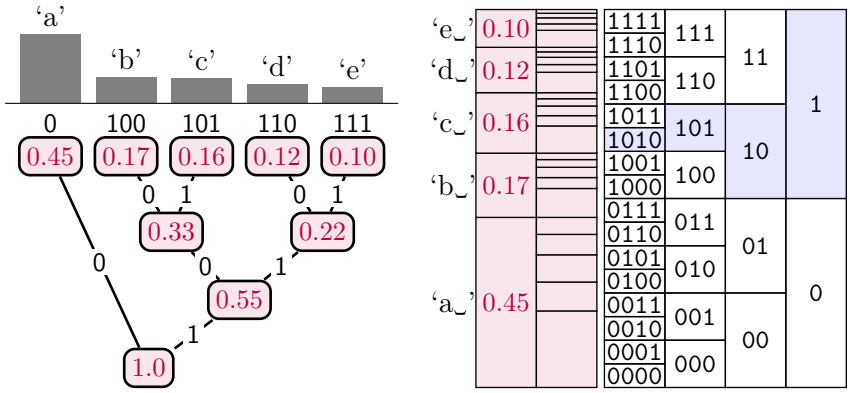


Figure 2.2: *Left:* A Huffman tree over five symbols. Purple numbers in boxes denote probabilities, while binary strings above the leaf nodes correspond to the codewords assigned by the Huffman algorithm. *Right:* Arithmetic coding of a message of length two. Each message corresponds to an interval whose length is proportional to the message's probability. As an example, any real number inside the interval corresponding to the message 'ca' begins with `0.1010...` when written in binary form so that we can use `1010` to uniquely encode it. The message 'aa' would be encoded as `000`.

Huffman coding assigns a codeword with length at most $\lceil -\log_2 P_X(x) \rceil$ for each symbol x, and can be shown to be optimal among all prefix codes [Cover and Thomas, 2006]. However, as log-probabilities are generally not integers, Huffman coding (as with all so-called *symbol codes*) incurs an overhead of up to 1 bit per symbol, relative to the information content. For example, a heavily biased coin with an entropy close to 0 will still require 1 bit to encode.

One way of reducing this overhead of symbol codes is by spreading the overhead over a group of symbols.[6] Given a message $(x_1, x_2, ...)$, we consider transmitting it in length-n blocks with a symbol code, effectively treating the source as i.i.d. length-n blocks with a new alphabet $\mathcal{X}^n$. The bound on the optimal prefix code length (Eq. (2.3)), averaged over each symbol, then becomes

$$\frac{1}{n}H[X_1, ..., X_n] \leq \frac{1}{n}L^*([X_1, ..., X_n]) \leq \frac{1}{n}H[X_1, ..., X_n] + \frac{1}{n}. \tag{2.7}$$

[6]Indeed, Shannon [1948, Theorem 9] already used block coding and the law of large numbers to show that an i.i.d. source can be losslessly compressed with an average code length arbitrarily close to the entropy. However, it was not until decades later with the invention of arithmetic coding that an algorithm made this idea a practical reality.

Thus, the 1 bit overhead on the RHS of (2.3) is reduced to $\frac{1}{n}$ and can be made arbitrarily low by increasing n. Note that in the usual case where $X_1, X_2, ...$ are i.i.d., the joint entropy factorizes so that $\frac{1}{n}H[X_1, ..., X_n] = H[X]$; otherwise, this block coding scheme can exploit the redundancy between the X_is and do better than coding one symbol at a time, since

$$H[X_1, ..., X_n] \leq \sum_{j=1}^{n} H[X_j],$$

for any n random objects $(X_1, ..., X_n)$.[7]

Despite the theoretically improved coding cost, directly applying a symbol code such as Huffman code over long blocks is often computationally infeasible, as the size of the requisite PMF table grows exponentially with the dimension of the block. It turns out that *stream codes* [MacKay, 2003, Chapter 6] to be discussed below will allow us to implement the idea behind block coding much more efficiently, while also offering a convenient mechanism for adapting our model to the source (we discussed modeling in Section 2.2.2) *during* compression.

2.2.4 Arithmetic Coding

Arithmetic coding [Rissanen and Langdon, 1979], also known as range coding [Martín, 1979], is an example of a *stream code* [MacKay, 2003, Chapter 6]. Stream codes differ from symbol codes in that they assign codewords to entire messages and individual symbols do not have unique codewords. Stream codes amortize the coding cost's overhead across the whole sequence of symbols and are therefore able to get closer to the entropy.

The basic idea of arithmetic coding is to associate each symbol with a subinterval of the interval $[0, 1]$ such that the subinterval's length equals the symbol's probability. This procedure can be generalized towards encoding symbols in sequence. For two symbols, the second

[7] In fact, by taking $n \to \infty$, the expression $\frac{1}{n}H[X_1, ..., X_n]$ defines the *entropy rate* of the process, which is the generalization of entropy to the case of a non i.i.d. source.

symbol is no longer considered a subinterval of $[0, 1]$ but a subinterval of the previous symbol's interval such that the second subinterval's length equals the two symbols' joint probability. For example, if symbol "a" has probability $1/5$ and is associated with $[0, 0.2)$, the sequence "aa" would be encoded as $[0, 0.04)$. This procedure can be iterated for sequences of arbitrary length, leading to a sequence of subintervals of decreasing size that contain each other. Any real-valued number contained in the final subinterval allows us to uniquely reconstruct the sequence of symbols and therefore encodes the entire sequence (Figure 2.2). In practice, one picks a representative number which can be represented with the smallest number of bits while uniquely identifying the interval. By construction, the size of the final sub-interval is the product of all previous symbols' conditional probabilities, hence the probability of the sequence. Intuitively, the interval's length (i.e., the sequence's joint probability) determines the number of relevant digits of the number representing the interval, thus the code length equals the log-probability of the sequence as required for entropy codes. If $x^n = (x_1, x_2, ..., x_n)$ is a message of length n, and $\gamma(x^n)$ the codeword assigned by arithmetic coding using a probability model P_θ, then the assigned code length satisfies [Polyanskiy and Wu, 2022, Chapter 13.1],

$$-\log_2 P_\theta(x^n) \leq |\gamma(x^n)| < \lceil -\log_2 P_\theta(x^n) \rceil + 1.$$

Thus if the true distribution of messages is P_{X^n}, then the average code length is at most $H[P_{X^n}, P_\theta] + 2$. The excess bits needed for encoding the message are amortized over all n symbols, making arithmetic coding more efficient on long messages than symbol codes. Per symbol, the overhead is only $2/n$ bits, compared to 1 bit per symbol for Huffman coding.

In arithmetic coding, the probability model $P_\theta(x^n)$ is only specified sequentially via the marginal $P_\theta(x_1)$ and conditional distributions $\{P_\theta(x_t|x_{<t}) : t = 2, ..., n\}$, thereby avoiding the exponential complexity of block coding (see discussion in the previous subsection). Moreover, a simple modification allows arithmetic coding to adapt its probability model "on the fly"

on the data sequence being compressed. I.e., when encoding a sequence x^n, the coding distribution $P_\theta(x_t|x_{<t})$ used at each time step t can be sequentially updated based on the encoded sequence $x_{<t}$ so far, for $t = 2, ..., n$; during decoding, the exact same sequence of model updates are performed based on the already received $x_{<t}$, so that the exact same model $P_\theta(x_t|x_{<t})$ is used for decoding as is for encoding. Adaptive arithmetic coding via sequential density estimation is thus an example of a *universal* source code [Cover and Thomas, 2006, Chapter 13], in the sense that the method can be applied without prior knowledge of the source distribution while still being asymptotically optimal for a large class of sources (e.g., all finite-order Markov chains).

Notably, arithmetic coding implements a first-in-first-out data structure, that is, a queue. Symbols which are encoded first are also decoded first. As we will see, this makes it a convenient choice for compression with autoregressive models (Section 3.1.2).

2.3 Lossy Compression

In the following we review *lossy* compression, that is, compression with imperfect data reconstruction. When working with continuous data, such as an analogue signal, lossy compression is necessary, as it is impossible to losslessly store real values with a finite number of bits. In other cases, a lossy reconstruction is often also sufficient, and the flexibility to discard "irrelevant" information allows us to achieve a far lower bit-rate than possible with lossless compression. Lossy compression algorithms for digital media, such as audio, images, and video, routinely obtain an order of magnitude or more bit-rate savings than their lossless counterparts, without noticeable loss of quality to the end users.

For our purpose, we formalize a lossy compression algorithm (or a lossy *codec*) as a 3-tuple consisting of an encoder, a decoder, and an entropy code, denoted by $c = (\mathtt{e}, \mathtt{d}, \gamma)$. The

encoder $\mathtt{e} : \mathcal{X} \to \mathcal{W}$ maps each data point $x \in \mathcal{X}$ to a representation w in a countable set $\mathcal{W}$ (this can be the set of natural numbers, without loss of generality). The *decoder* $\mathtt{d} : \mathcal{W} \to \hat{\mathcal{X}}$ maps each representation w to a point in the *reproduction* space (or, reproduction alphabet) $\hat{\mathcal{X}}$, which needs not be the same as $\mathcal{X}$. The encoder and decoder further agree to transmit the discrete representation $W = \mathtt{e}(X)$ using an entropy code $\gamma : \mathcal{W} \to \{0,1\}^*$ (see Section 2.2); we write $l(x) := |\gamma(\mathtt{e}(x))|$ to denote the code length assigned to the encoding of x. Conceptually, a lossy codec defines a Markov chain

$$X \xrightarrow{\mathtt{e}} W \xrightarrow{\mathtt{d}} \hat{X}.$$

Suppose we measure the reconstruction error between a source and reproduction point by a suitable *distortion function* $\rho : \mathcal{X} \times \hat{\mathcal{X}} \to [0, \infty)$. Historically, this is typically a squared error, i.e., $\rho(x, y) \propto \|x - y\|^2$ in the continuous case, or Hamming distance in the discrete case. Given the above, lossy compression is generally concerned with minimizing the average *distortion*

$$\mathcal{D} = \mathbb{E}[\rho(X, \hat{X})], \tag{2.8}$$

where $\hat{X} := \mathtt{d}(\mathtt{e}(X))$, and simultaneously, the *(bit-)rate*

$$\mathcal{R} = \mathbb{E}[l(X)], \tag{2.9}$$

that is, the average number of bits needed to encode a data sample. We want to minimize these two quantities with respect to the choice of the encoding and decoding procedures $\mathtt{e}$ and $\mathtt{d}$, as well as the entropy code γ. Generally, minimizing rate comes at the expense of increased distortion, and vice versa, and a trade-off has to be made.

2.3.1 Rate-Distortion Theory

Rate-distortion (R-D) theory [Shannon, 1959, Cover and Thomas, 2006] formally studies the fundamental trade-off between the average number of bits per sample (*rate*) used to represent the source X and the *distortion* incurred by the lossy representation $\hat{X}$. It asks the following question about the limit of lossy compression: "for a given data source and a distortion metric, what is the minimum number of bits (per sample) needed to represent the source at a tolerable level of distortion, regardless of the computation complexity of the compression procedure?" The answer turns out to be given by the information rate-distortion function [Shannon, 1959], which, for an i.i.d. source, is defined as follows,

$$R_I(D) = \inf_{Q_{\hat{X}|X}:\ \mathbb{E}[\rho(X,\hat{X})]\leq D} I(X;\hat{X}), \tag{2.10}$$

where we consider all random transforms $Q_{\hat{X}|X}$ whose expected distortion is within the given threshold $D \geq 0$, and minimize the mutual information between the source X and its reproduction $\hat{X}$. The mutual information $I(X;\hat{X})$ is defined as the relative entropy (i.e., KL divergence) between the $P_X \otimes Q_{\hat{X}|X}$ and the product of its two marginal distributions. [8]

Shannon's lossy source coding theorems [Shannon, 1959] gave operational significance to the above mathematical definition: for a given $D \geq 0$, $R(D)$ is the minimum achievable bits/sample rate with which any lossy compression algorithm can represent the source with an average distortion less than or equal to D. The (information) R-D function is a fundamental quantity that depends only on the source and choice of distortion function, and generalizes the Shannon entropy H (Eq. 2.2) from lossless compression to lossy compression. The R-D function has a few general properties, e.g., it is always non-increasing and convex, but is otherwise unknown analytically outside of a few cases. A well-known algorithm by Blahut

[8] We use the following general definition of relative entropy [Polyanskiy and Wu, 2022, Definition 2.1]: for two measures α, β defined on a common measurable space, $H(\alpha|\beta) := \int \log(\frac{d\alpha}{d\beta}) d\alpha$ when α is absolutely continuous w.r.t β, and infinite otherwise.

[Blahut, 1972] and [Arimoto, 1972] can be used to compute $R(D)$ in the case of a known discrete source in low dimension, and Part II of this thesis is devoted to estimating $R(D)$ for more general and high-dimensional sources such as images and speech.

In theory, the optimal rate $R_I(D)$ is achievable by vector quantization [Cover and Thomas, 2006], to be described in Section 2.3.2, but this approach to compression quickly becomes intractable for high-dimensional data. Rather than trying to achieve the theoretically optimal rate $R_I(D)$ with any codec at any computational cost, in practice the design of a lossy codec is constrained by practical considerations, such as the computation budget of the target hardware, or the decoding latency acceptable for the application. Denoting the set of all the acceptable codecs under consideration by $\mathcal{C}$, we can instead consider an *operational* rate-distortion function [9], formalized by

$$R_O(D) = \inf_{c \in \mathcal{C}: \mathbb{E}[\rho(X,\hat{X})] \leq D} \mathbb{E}[l(X)]. \tag{2.11}$$

Compared to Eq. 2.10, we replaced mutual information by the operational rate and optimize over an actual lossy codec c. We can relax the constrained optimization problem to an unconstrained one, by introducing the rate-distortion Lagrangian [Blahut, 1972, Chou et al., 1989],

$$L(\lambda, c) = \mathcal{R}(c) + \lambda \mathcal{D}(c) = \mathbb{E}[l(X)] + \lambda \mathbb{E}[\rho(X, \hat{X})]. \tag{2.12}$$

For each fixed $\lambda > 0$, the minimum of this objective yields a codec c^* whose operational distortion-rate performance, $(\mathcal{D}(c^*), \mathcal{R}(c^*))$, lies on the convex hull of the operational R-D

[9] In usual treatments of R-D theory (see e.g., [Cover and Thomas, 2006]), *the* operational R-D function is defined as the asymptotic rate achievable by compressing multiple samples jointly using *any* lossy codec, in the limit of infinitely many jointly compressed samples. This is different from our definition here, which instead focuses on the operational R-D performance achievable within a constrained family of codecs $\mathcal{C}$, by compressing a single sample at a time (as is common in applications). Thus we generally only have $R_O(D) \geq R_I(D)$, i.e., $R_I(D)$ is in general no longer achievable in our "one-shot" setup.

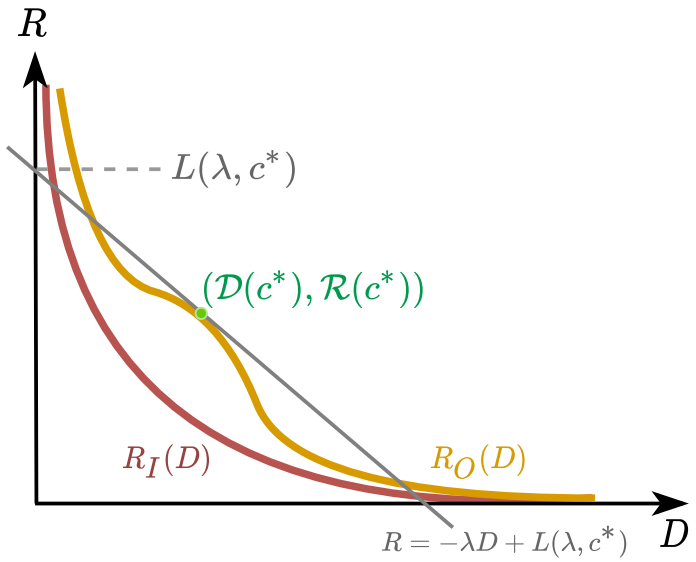


Figure 2.3: Visualizing the operational R-D optimization problem. We adjust our codec c to minimize the R-axis intercept of a straight line (gray) with slope $-\lambda$ and passing through $(\mathcal{D}(c), \mathcal{R}(c))$; an optimum occurs when the line becomes tangent to the operational R-D curve R_O at the point $(\mathcal{D}(c^*), \mathcal{R}(c^*))$. Note that the operational R-D curve (orange) is not necessarily convex, and always lies above the information R-D curve R_I (red), i.e., $R_O(D) \geq R_I(D), \forall D$.

curve.[10] This is illustrated in Figure 2.3. The hyperparameter λ can be interpreted as a Lagrange multiplier, and codecs with different operational rate-distortion trade-offs can be found by minimizing the Lagrangian with various λ. Most current end-to-end learned lossy compression methods to be discussed in Section 3.2 follow this approach, training one codec for each λ. We note that it is not always possible to attain every point on the operational R-D curve with this approach [Degrave and Korshunova], and alternative methods based on constrained optimization have been proposed [Platt and Barr, 1988, van Rozendaal et al., 2020].

A word on notation: the rest of the chapter deals with a vector-valued source (typically $\mathcal{X} = \mathbb{R}^N$), and uses bold letters $\mathbf{x}, \mathbf{X}$ for emphasis.

[10] In practice, the optimization problem is often non-convex in the parameters of the codec, so we typically reach a local minimum of the Lagrangian, corresponding to an R-D point lying above the operational R-D curve.

2.3.2 Vector Quantization

Vector quantization (VQ), a classical technique from signal processing, is perhaps the most basic and general form of a lossy codec. A vector quantization scheme, or a vector quantizer, consists of a set of integer representations $\mathcal{W} = \{1, 2, ..., k\}$, an encoder $\texttt{e}$ assigning a given data to an integer representation $w \in \mathcal{W}$, and a decoder $\texttt{d}$ that returns a reconstruction given w, usually implemented by indexing into a codebook of k reconstruction vectors. Note that there is no restriction on the encoding and decoding functions, other than the cardinality of $\mathcal{W}$. The goal is then to determine an optimal quantization scheme for a given data distribution, under some objective. Commonly, the objective is to minimize a reconstruction error (Eq. 2.8), as in the k-means algorithm, but can also more generally be an operational rate-distortion trade-off (Eq. 2.11). An optimal quantizer can be shown to always encode any given data point to its "nearest-neighbor" in the codebook to minimize the reconstruction error ρ (or, a combination of reconstruction error and code length, in the rate-constrained version [Chou et al., 1989]), and set the jth entry in the codebook $\hat{\mathbf{x}}^{(j)}$ to minimize the expected reconstruction error between $\hat{\mathbf{x}}^{(j)}$ and data points assigned to index j [Chou et al., 1989].

For some data source, e.g., the uniform or the Laplace distribution, optimal quantization can be characterized analytically [Ziv, 1985, Sullivan, 1996]. In most applications, Lloyd-Max-style algorithms (more widely known as k-means in machine learning) [Lloyd, 1982][Chou et al., 1989] are instead used to estimate a quantization scheme from data samples. This usually involves minimizing an empirical rate-distortion cost (Eq. 2.12) over a dataset, which is still a basic ingredient in today's learned compression approaches [Ballé et al., 2021] (see Sec. 3.2).

Given unlimited data and compute, VQ can approximate any data distribution arbitrarily well. Indeed, the theoretical limit of lossy compression, the rate-distortion function $R_I(D)$, can be shown to be achievable by jointly quantizing multiple data samples together with

increasingly long blocks [Cover and Thomas, 2006]. However, VQ comes with the severe downside of poor scalability and data efficiency [Gersho and Gray, 2012]. The computational and storage demand of VQ increases quickly with the data dimensionality. In high dimensions, an exceedingly large number of quantization points and high amounts of training data are needed to approximate the data distribution well. As a result, VQ is typically found in low-rate applications, such as low-rate speech coding [Sayood, 2012]. We note however, k-means-style vector quantization has been successfully applied in a convolutional latent space for image generation [Oord et al., 2017], and serves as an image tokenizer in many recent transformer-based models for image [Esser et al., 2021, Yu et al., 2021] and text-to-image generation [Ramesh et al., 2021, Yu et al., 2022].

2.3.3 Transform Coding

Instead of quantizing the data directly, it is often much easier to do so in a transformed space, where the representation of the data is uncorrelated and scalar quantization can be effectively applied. The core idea behind transform coding [Goyal, 2001] is therefore to divide the task of lossy compression into decorrelation and quantization: first, the sender applies an *analysis transform* f to data $\mathbf{x}$, resulting in (ideally decorrelated) transform coefficients, $\mathbf{z} = f(\mathbf{x})$; and second, coordinate-wise scalar quantization $[\![\cdot]\!]$ is applied to obtain a discretized representation $\hat{\mathbf{z}} = [\![\mathbf{z}]\!]$. The symbols representing $\hat{\mathbf{z}}$ can then be converted to a bit-string by entropy coding (discussed in Section 2.2) under an entropy model $P(\hat{\mathbf{z}})$. The receiver losslessly recovers $\hat{\mathbf{z}}$, and computes a reconstruction $\hat{\mathbf{x}} = g(\hat{\mathbf{z}})$ using a *synthesis transform* g, which is often the inverse of the analysis transform. In the terminology of Section 2.3, the encoder $\mathtt{e} = [\![\cdot]\!] \circ f$ is the composition of the analysis transform f followed by quantization, and the decoder $\mathtt{d}$ is implemented by the synthesis transform g.

The analysis transforms is typically linear and invertible, and is implemented as multiplication

with an orthogonal matrix. A classic example is the Karhunen-Loève Transform (KLT), which is the same as principle component analysis (PCA) in our setting. The KLT maps data samples into the eigen-basis of the covariance matrix of the data distribution, and produces uncorrelated transform coefficients $\mathbf{z}$ [Gersho and Gray, 2012, Section 8.6]. Another example is the discrete cosine transform (DCT), which can be shown to approximate PCA asymptotically [Ahmed et al., 1974], and is one of the most widely used transforms in signal processing and compression (e.g., in JPEG). The choice of an orthogonal transform simplifies the theoretical analysis and design of transform coding algorithms, and for Gaussian data, the KLT followed by uniform scalar quantization can be shown to be optimal [Goyal et al., 2000].

Unlike vector quantization, which effectively optimizes over all possible choices of encoder $\mathtt{e} : \mathcal{X} \to \mathcal{W}$ and decoder $\mathtt{d} : \mathcal{W} \to \hat{\mathcal{X}}$, transform coding implicitly restricts the solution space of allowable codecs, e.g., the set of quantization points must be in the range of g, and therefore generally cannot achieve the unconstrained optimal performance of VQ. The lack of theoretical optimality of transform coding is more than made up for by its vastly superior scalability over VQ, as evidenced by its wide-spread use in the compression of digital media, such as images and videos [Goyal, 2001].

Chapter 3

Background: Neural Compression

Neural compression is the application of neural networks and deep learning to compression. While statistical estimation has always been a core aspect of compression, deep learning brings in new tools and possibilities to compression. Building on the basic concepts of source coding in Chapter 2, we now introduce some of the main approaches to neural compression.

As a side note, deep learning has been applied to many problems that can be considered parts of a data compression pipeline; for example, image compression artifact removal [Galteri et al., 2017, Kawar et al., 2022], or various pre- and post-processing stages of video codecs [Ma et al., 2019, Duong et al., 2023]. Here our focus is on neural methods that are directly optimized end-to-end for a compression cost, e.g., bit-rate and/or reconstruction quality.

The common background required for the thesis is non-linear transform coding (Section 3.2.1), which mainly builds on Chapter 2 and the discretized density model (Section 3.1.1). Chapter 4 additionally builds on bits-back coding (Section 3.1.3), and Chapter 6 also references the basic connection between latent variable modeling and bits-back coding (Section 3.1.3). Chapters 5, 6, and 7 make brief references to the idea of relative entropy coding (Section 3.2.2).

A word on notation: unlike in Chapter 2 which considers an abstract source X, this chapter follows the machine learning convention and uses bold symbols $\mathbf{x}, \mathbf{z}, \mathbf{X}, ...$ to emphasize that the denoted quantities are vectors (rather than scalars x, z, X).

We use P_θ to denote a probability model, which is generally a probability measure on the data space $\mathcal{X}$. In the discrete case, P_θ also double-duties as its probability mass function (PMF). We use p_θ to generally denote a probability density model, which can mean either a probability mass function or probability density function depending on context.

Much of the material in this chapter has been published in the following research monograph,

An Introduction to Neural Data Compression. Yibo Yang, Stephan Mandt, and Lucas Theis. *Foundations and Trends in Computer Graphics and Vision, 2023*,

which has a more comprehensive treatment of neural compression.

3.1 Neural Lossless Compression

Most of the neural loss compression methods follow a two-step recipe: 1. estimate a probability model p_θ of the data; 2. convert any given data $\mathbf{x}$ to a bit-string, using the model together with an entropy coding algorithm such as arithmetic coding. The first step requires us to develop an accurate probability model of discrete data, potentially of very high dimensionality (e.g., a 64×64 RGB image has over ten thousand dimensions). The models we will discuss often take a divide-and-conquer approach, reducing the problem of modeling high-dimensional data into a collection of univariate density estimation problems. Once we obtained a probability model p_θ, we can then invoke a suitable entropy coding algorithm to assign a bit-string to any new data $\mathbf{x}$ according to the model. This step is usually straightforward, although

sometimes ingenuity is required, e.g., in the case of a latent variable model. Thus neural lossless compression can largely be viewed as an extension of density estimation for discrete data, with an emphasis towards practical compression algorithms. There is typically a trade-off between the compression performance (bit-rate) and the computation complexity of encoding/decoding.

3.1.1 Continuous Models for Discrete Data

Before proceeding further, we discuss how a model for discrete data can be defined and parameterized by a continuous model, a common technique in generative modeling and neural compression. Lossless compression generally operates on discrete data. After all, it is infeasible to losslessly encode any real number by a finite number of bits. However, many generative models in machine learning assume continuous data, and model it with a density function. It turns out such continuous models are still useful for lossless compression.

Suppose we specify a density model p_θ over $\mathbb{R}^N$, and $\mathbf{x} \in \{0, \dots, 255\}^N$ is an RGB image following the ground truth image distribution $P_{\mathbf{X}}$. We can derive a PMF over integers by integrating the density over hypercubes, as follows,

$$P_\theta(\mathbf{x}) := \int_{[-.5,.5)^N} p_\theta(\mathbf{x} + \mathbf{u})\, d\mathbf{u}. \tag{3.1}$$

Let $\mathbf{u} \sim \mathcal{U}([-0.5, 0.5)^N)$ be an independent uniform noise over the unit hypercube, and consider fitting the density model p_θ on the noisy (or *dequantized* [Uria et al., 2013]) data, $\mathbf{x} + \mathbf{u}$. Then it is not difficult to show that [Theis et al., 2016]

$$-\mathbb{E}_{\mathbf{x}\sim P_{\mathbf{X}}, \mathbf{u}\sim\mathcal{U}}[\log p_\theta(\mathbf{x} + \mathbf{u})] \geq -\mathbb{E}_{\mathbf{x}\sim P_{\mathbf{X}}}[\log P_\theta(\mathbf{x})]. \tag{3.2}$$

The left-hand side is the negative log-likelihood which we would optimize if we our density

model to the dequantized data. Thus, we can minimize an upper bound on the lossless compression cost under the discretized model (RHS) by fitting a density via maximum likelihood (LHS), provided the discrete data is *dequantized* appropriately with noise [Uria et al., 2013].[1]

The form of P_θ as defined in Eq. 3.1, which we call a *discretized density model*, is also useful in itself as a flexible model for discrete data. Examples include PixelCNN++ [Salimans et al., 2017], the prior distribution in a discrete flow [Hoogeboom et al., 2019], as well as entropy models for neural compression [Mentzer et al., 2019, Ballé et al., 2018a, Minnen et al., 2018].

The integral in Eq. (3.1) in general is intractable to compute, but it can often be broken down into a series of univariate integrals. Therefore let us consider a univariate version of the discretized density model,

$$P_\theta(x) := \int_{[-0.5,0.5)} p_\theta(x+u)du, \qquad x \in \mathbb{N}.$$

Let F_θ denote the CDF of p_θ, then we can equivalently express the above in terms of a difference of CDF evaluations:

$$P_\theta(x) := F_\theta(x+0.5) - F_\theta(x-0.5). \tag{3.3}$$

For example, if p_θ is the density of a logistic distribution, then F_θ is the logistic sigmoid function common in deep learning and has an efficient numeric implementation.

[1] This approach of adding dequantization noise is commonly used for training continuous density models on discrete data, and can be extended to optimize a tighter upper bound on the RHS via *variational dequantization* [Ho et al., 2019b, Hoogeboom et al., 2021b].

3.1.2 Compression with Autoregressive Models

Autoregressive models exploit the fact that we can write any probability distribution as a product of conditional distributions using the chain rule of probabilities [Bishop, 2006], [2]

$$p(\mathbf{x}) = \prod_i p(x_i \mid \mathbf{x}_{<i}). \tag{3.4}$$

Here, x_i is the i-th entry of the vector $\mathbf{x}$ and $\mathbf{x}_{<i}$ corresponds to all previous entries in an arbitrary order. The autoregressive factorization does not make any assumptions about the data distribution yet still allows us to easily incorporate useful assumptions. For example, a Markov assumption can be implemented by only considering the entries in $\mathbf{x}_{<i}$ which are close to i. A stationarity assumption is easily incorporated by using the same conditional distribution $p(x_i \,|\, \mathbf{x}_{<i})$ at every location. These two assumptions are often reasonable and can drastically reduce the amount of parameters of a model.

Autoregressive modeling lends itself to lossless compression in combination with arithmetic coding (Section 2.2.4) since both deal with data sequentially, one symbol at a time. Each symbol is encoded using the conditional distribution given the data that has already been encoded. This is in contrast to Markov random fields, for example, where conditional distributions are typically only tractable when conditioning on a larger neighborhood. Entropy coding generally requires the number of symbols in each encoding step to be manageable. Autoregressive models provide an important step towards practical entropy coding by decomposing a high-dimensional distribution into low-dimensional conditional distributions.

For data modalities such as audio or text, autoregressive models are an obvious choice due to the signal's sequential nature. Indeed, two of the top performing algorithms in the Large Text Compression Benchmark [Mahoney] use recurrent neural networks to predict the next

[2] To lighten notation, we drop θ from the model distribution $p_\theta(\mathbf{x})$, and similarly in the conditionals $p_\theta(x_i \mid \mathbf{x}_{<i})$.

token of a sentence. While `cmix` [Knoll] uses a mixture of long short-term memory networks (LSTMs) [Hochreiter and Schmidhuber, 1997] and nonparametric models, `nncp` [Bellard, 2019] relies solely on neural network models. However, autoregressive models have long also found application in image compression. For example, JPEG [ITU-T, 1992] encodes the difference between neighboring DC coefficients, $z_{ij}^{\mathrm{DC}} - z_{i(j-1)}^{\mathrm{DC}}$, in a raster-scan order, which can be thought of as implementing a first-order Markov model.

Mixtures of experts [Jacobs et al., 1991, Hosseini et al., 2010, Theis et al., 2012] are a class of autoregressive models proven useful for compressing images,

$$p(x_{ij} \mid \mathbf{x}_{<ij}) = \sum_k \underbrace{p(k \mid \mathbf{x}_{<ij})}_{\text{Gates}} \underbrace{p(x_{ij} \mid \mathbf{x}_{<ij}, k)}_{\text{Experts}}. \tag{3.5}$$

The basic idea behind neural network extensions of this approach is to nonlinearly transform the inputs $\mathbf{x}_{<ij}$ before feeding them into the gates and the experts. For instance, RNADE [Uria et al., 2013] used a fully connected neural network with a single rectified linear layer to transform the inputs $\mathbf{x}_{<ij}$. Other examples of deep autoregressive models include RIDE [Theis and Bethge, 2015], PixelRNN [van den Oord et al., 2016], or PixelCNN++ [Salimans et al., 2017]. More recent autoregressive modeling papers continue to explore different architectures for transforming $\mathbf{x}_{<ij}$. For example, the Image Transformer [Parmar et al., 2018] uses an architecture based on self-attention [Vaswani et al., 2017]. PixelSNAIL [Chen et al., 2018a] uses a combination of convolutions and self-attention layers. Glow [Kingma and Dhariwal, 2018] uses invertible flows.

With the exception of `nncp` [Bellard, 2019] and `cmix` [Knoll], all autoregressive models mentioned so far are *static* models. That is, the models' parameters are trained once and then remain fixed during the entire encoding and decoding process. In contrast, a *dynamic* or *adaptive* model updates its parameters during the encoding process based on already encoded data. The decoder is then able to apply the same model updates based on the data already

received.[3] A simple dynamic autoregressive model for images was proposed by Wu et al. [1998]. Here, the predictors are linear but the predictor's parameters are chosen dynamically. This is done by treating a larger neighborhood of preceding pixels as training data for the predictor. Meyer and Tischer [2001] improved on this idea by weighting training points based on their distance to the pixel whose value we are predicting.

Autoregressive models come with the restriction that decoding is an inherently sequential procedure. Each symbol can only be decoded after all the symbols have been decoded on which its prediction depends. To improve decoding speed, we can restrict the context $\mathbf{x}_{<i}$ to only a small neighborhood around x_i, as for example in JPEG. Furthermore, the degree of parallelism can be increased by grouping coordinates of the data into blocks and only modeling the conditional dependencies between blocks, while treating data coordinates within each block as conditionally independent [Stern et al., 2018, Minnen and Singh, 2020].

3.1.3 Latent Variable Models and Bits-back Coding

Latent variable models represent the data distribution using the sum rule of probability,

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) \, \mathrm{d}\mathbf{z} = \int p(\mathbf{x} \mid \mathbf{z}) p(\mathbf{z}) \, \mathrm{d}\mathbf{z}, \tag{3.6}$$

where $\mathbf{z}$ is a vector of *latent variables* (also called "latents"), and the joint distribution $p(\mathbf{x}, \mathbf{z})$ factorizes into a *prior* $p(\mathbf{z})$ and a *likelihood* $p(\mathbf{x} \,|\, \mathbf{z})$. Latent variable models are ubiquitous in machine learning and include hidden Markov models, mixture models [Bishop, 2006], and more recently variational autoencoders (VAEs) [Kingma and Welling, 2014, Rezende et al., 2014a]. By integrating or summing over all possible realizations of the latent vector, latent variable models can capture complex dependencies in the data even when the prior and likelihood take simple forms.

[3] We already introduced this idea when discussing adaptive Arithmetic Coding in Section 2.2.4.

Training a latent variable model by (approximate) maximum-likelihood (see Eq. (2.6)) comes with a challenge: unlike in a fully-observed (e.g., autoregressive) model where the data probability $p(\mathbf{x})$ can be readily evaluated, doing so is no longer straightforward since it is defined through an often intractable integral (Eq. 3.6). Variational inference deals with exactly these complications, and we refer to [Zhang et al., 2018] for more details on these methods. Here, we instead focus on the compression problem, which faces a similar challenge and uses similar tools. Given a latent variable model consisting of a prior $p(\mathbf{z})$ and likelihood $p(\mathbf{x}|\mathbf{z})$, our goal is to compress a given data vector $\mathbf{x}$ with a code length close to its information content under the model, $-\log_2 p(\mathbf{x})$. To simplify the discussion below, we assume discrete $\mathbf{z}$ unless specified otherwise.

Two-Part Code

We start by considering a simple although generally suboptimal *two-part code* [Grünwald, 2007]. Here, the data $\mathbf{x}$ is transmitted along with a latent vector $\mathbf{z}$ in two steps. First, the sender decides on a (ideally informative) vector $\mathbf{z}$, then encodes and transmits it under the prior $p(\mathbf{z})$. Next, $\mathbf{x}$ is compressed and transmitted under the likelihood model, $p(\mathbf{x} \,|\, \mathbf{z})$. The receiver then decodes $\mathbf{z}$, and finally $\mathbf{x}$ given $\mathbf{z}$, using the same models. Both the prior and likelihood models are often fully factorized, allowing for coding each dimension of $\mathbf{z}$ or $\mathbf{x}$ in parallel, but can also be autoregressive models used in conjunction with arithmetic coding. Assuming negligible entropy coding overhead, the combined code length is

$$l(\mathbf{x}, \mathbf{z}) = -\log_2 p(\mathbf{z}) - \log_2 p(\mathbf{x} \mid \mathbf{z}) = -\log_2 p(\mathbf{x}, \mathbf{z}). \tag{3.7}$$

Moreover, the sender can (at least in principle) minimize this quantity over all choices of $\mathbf{z}$, resulting in the code length

$$l_{\mathrm{MTP}}(\mathbf{x}) = \min_{\mathbf{z}} \left(-\log_2 p(\mathbf{z}) - \log_2 p(\mathbf{x} \mid \mathbf{z})\right). \tag{3.8}$$

The minimal two-part code length, $l_{\mathrm{MTP}}(\mathbf{x})$, is generally still suboptimal [Frey, 1998, Wallace, 1990, Honkela and Valpola, 2004].[4] To see this, consider the following bound on the information content $-\log_2 p(\mathbf{x})$,

$$-\log_2 p(\mathbf{x}) \leq -\log_2 p(\mathbf{x}) + D_{\mathrm{KL}}(q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z} \mid \mathbf{x})) \tag{3.9}$$

$$= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}[-\log_2 p(\mathbf{z}, \mathbf{x})] - H[q(\mathbf{z} \mid \mathbf{x})], \tag{3.10}$$

also known as the *negative evidence lower bound* (NELBO) of variational inference [Zhang et al., 2018]. Here, q is any distribution over $\mathbf{z}$ which may depend on $\mathbf{x}$. Crucially, the minimal two-part code length (Eq. 3.8) is a special case of the NELBO where

$$q(\mathbf{z} \mid \mathbf{x}) = \delta_{\mathbf{z}_{\min}}(\mathbf{z}) \tag{3.11}$$

is a degenerate distribution centered on $\mathbf{z}_{\min}$, which achieves the minimum in Eq. 3.8. In this case the entropy term vanishes. More generally, the NELBO is minimized when $q(\mathbf{z} \mid \mathbf{x})$ equals the Bayesian *posterior* distribution $p(\mathbf{z} \mid \mathbf{x})$, the target of approximate inference [Zhang et al., 2018].

One may naturally wonder whether a coding cost equal to the NELBO can be realized for *any choice of* $q(\mathbf{z} \mid \mathbf{x})$. Bits-back coding, to be discussed next, answers this question in the affirmative. It further offers a compression interpretation of variational inference: for a

[4] However, it is possible to directly optimize a neural compression method for the two-part code. e.g., Mentzer et al. [2019] trained a latent variable model with an inference network to minimize the expected coding cost of the two-part code, demonstrating much faster decoding speed than autoregressive models, although at $\sim 20\%$ worse bit-rate.

given latent variable model, reducing the KL divergence between the distribution q and the posterior distribution is equivalent to minimizing the code length of $\mathbf{x}$ under bits-back coding.

Bits-Back Coding

While the optimal two-part code chooses a best latent vector $\mathbf{z}$ using deterministic optimization, bits-back coding [Frey and Hinton, 1997, Wallace, 1990, Hinton and Van Camp, 1993, Honkela and Valpola, 2004] instead uses a *stochastic* latent code in the form of a sample $\mathbf{z}$ from a distribution q. Surprisingly, this stochastic code can save bits compared to a deterministic code by allowing auxiliary information to piggyback on the choice of the latent code. To gain some intuition, consider again the optimization in Eq. 3.8 for picking the optimal two-part code. Given data $\mathbf{x}$, suppose there is some ambiguity about the optimal value of $\mathbf{z}$. For the sake of argument, suppose the minimization over $\mathbf{z}$ finds m equally optimal candidates $\{\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, ..., \mathbf{z}^{(m)}\}$. Instead of choosing one such $\mathbf{z}^{(i)}$ arbitrarily, the encoder can do so intentionally so as to communicate $\log m$ bits of additional information via the chosen integer index, provided that the decoder can reverse-engineer the encoding procedure.

Before proceeding, we emphasize a connection between sampling and decoding random bits. Given an entropy code derived from a discrete distribution, using it to *decode* a sequence of uniformly random bits produces a random sample from this distribution. For example, consider decoding a random bit-string according to a Huffman tree (Figure 2.2). As we traverse from the root to any leaf node, every edge along the path corresponds to a decision according to a coin flip. The probability of ending up with a particular symbol $\mathbf{x}$ with code length $|C(\mathbf{x})|$ is $2^{-|C(\mathbf{x})|}$, or approximately $2^{\log_2 p(\mathbf{x})} = p(\mathbf{x})$.

In bits-back coding, the sender first generates a sample $\mathbf{z} \sim q(\mathbf{z} \,|\, \mathbf{x})$ by *decoding* a random bit-string ξ. Next, the sender encodes $\mathbf{x}$ under $p(\mathbf{x} \,|\, \mathbf{z})$, then $\mathbf{z}$ under $p(\mathbf{z})$, and then transmits the resulting bits. The receiver begins by decoding $\mathbf{z}$ and $\mathbf{x}$ from the received bits just as

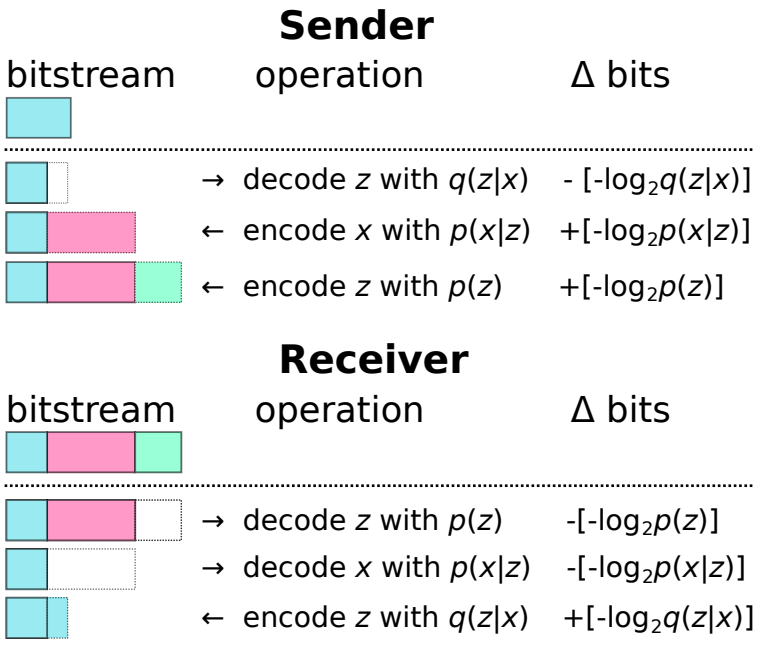


Figure 3.1: An illustration of the encoding (sender) and decoding (receiver) operations of bits-back coding. Auxiliary bits (ξ) are colored in blue.

in the two-part code, but then proceeds to recover the exact bit-string the sender used to generate $\mathbf{z}$. This can be done by *encoding* $\mathbf{z}$ under the distribution $q(\mathbf{z} \mid \mathbf{x})$, which we assume the receiver has access to. Thus, $-\log_2 q(\mathbf{z}|\mathbf{x})$ extra bits of information are recovered by the decoder (in addition to $\mathbf{x}$) "free of charge", giving the scheme its name "bits-back" coding. Finally, and crucially, we note that the initial bits used to generate $\mathbf{z}$ do not have to be truly random. Indeed, to be useful, they should be supplied from a well-compressed bit stream of *auxiliary information* that also needs to be transmitted. For example, if our application requires us to transmit an image along with a thumbnail version of it (e.g., for fast preview), the initial bits can come from the bit-string of the thumbnail image encoded with JPEG. We will examine this issue in more detail shortly.

Figure 3.1 graphically summarizes the algorithm. Given a bit-string ξ of auxiliary bits and the data $\mathbf{x}$ to transmit, the sender starts by decoding a portion of ξ into a stochastic latent code $\mathbf{z}$ of the data, ultimately sending a total number of bits equal to $|\xi| - \log_2 p(\mathbf{z}) - \log_2 p(\mathbf{x}|\mathbf{z}) + \log_2 q(\mathbf{z}|\mathbf{x})$ to communicate both ξ and $\mathbf{x}$. The net number of bits used for transmitting $\mathbf{x}$

alone is therefore

$$-\log_2 p(\mathbf{z}) - \log_2 p(\mathbf{x}|\mathbf{z}) - (-\log_2 q(\mathbf{z}|\mathbf{x})) \tag{3.12}$$

which corresponds to a discount of $-\log_2 q(\mathbf{z}\,|\,\mathbf{x})$ bits compared to a two-part code. Since $\mathbf{z}$ was drawn from $q(\mathbf{z}|\mathbf{x})$, the expected coding cost is exactly the NELBO [Zhang et al., 2018].

The choice of auxiliary information is an important one in bits-back coding. If the sender only wishes to communicate a single data sample $\mathbf{x}$ and no additional auxiliary information, the sender may set ξ to be an artificial sequence of random bits, which can still be recovered by the receiver, but contains no useful information. In this case we do not get "bits back", and suffer an expected code length that is worse than the best two-part code. Even when there are auxiliary bits to transmit, the sender needs to commit to sending at least $-\log q(\mathbf{z}|\mathbf{x})$ many of them for bits-back coding to achieve its nominal efficiency. This is known as the *initial bits problem* [Frey, 1998, Townsend et al., 2019a]. One potential solution is to compress a small portion of the data (e.g., a part of an image) with another codec, and use the resulting compressed bit-string for auxiliary bits [Townsend et al., 2019b]. If multiple data samples need to be compressed, an elegant solution is *chaining* [Townsend et al., 2019a] or "bits-back with feedback" [Frey, 1998]. The idea is to use the running bitstream of previously encoded data samples $\mathbf{x}^{(1)}, ..., \mathbf{x}^{(i-1)}$ as the source of auxiliary bits for encoding the sample $\mathbf{x}^{(i)}$. The initial bits problem becomes less severe with more data samples, and the code length per sample asymptotically converges to the NELBO. Chaining poses a requirement on the entropy coder that data is decoded in the exact opposite order in which it is encoded. Bits-back coding is therefore naturally combined with the ANS algorithm [Duda, 2009], which can be seen as a faster version of arithmetic coding but operates in stack (instead of queue) order, popularized by the BB-ANS algorithm [Townsend et al., 2019a].

3.2 Neural Lossy Compression

Most neural lossy compression approaches follow the idea of transform coding – i.e., transforming the input data into a set of "latent" coefficients, which can then be quantized (or go through some other lossy operation) and modeled more simply than the original data, e.g., via scalar quantization and with a factorized entropy model. The main new idea is to parameterize such a transform with neural networks, and optimize the compression cost end-to-end on the target data. Non-neural approaches learned from data that directly operate in the data space are also possible; see recent work on Variational Entropy Constrained Vector Quantization (VECVQ) [Ballé et al., 2021] and the method of [Yang et al., 2023b] which learn vector quantizers that are optimal in the rate-distortion sense.

Compared to lossless compression, which focuses on lowering the bit-rate, lossy compression introduces additional criteria concerning the quality of the lossy reconstruction, such as distortion (w.r.t. the original data) [Shannon, 1959], realism or perceptual quality (which may be measured with or without reference to the original data) [Zhang et al., 2018, Blau and Michaeli, 2018, Blau and Michaeli, 2019], or some custom-defined notion of utility such as performance on downstream tasks [Singh et al., 2020, Dubois et al., 2021]. Compared to traditional codecs, neural compression has the distinct advantage that it can be learned end-to-end on such objectives of interest. The discussion of some of these criteria can be quite involved and are active areas of research, and touch on subjects including signal quality assessment [Eskicioglu and Fisher, 1995, ITU-T, 2016], psychophysics of the human visual system [Zhou Wang et al., 2004, Wang et al., 2003], or even algorithmic information theory [Theis, 2024]. Here we will focus on the *distortion* criterion [Shannon, 1959] that has traditionally received the most attention, and refer to [Yang et al., 2023a] for discussions involving the other criteria.

3.2.1 Nonlinear Transform Coding

Nonlinear transform coding (NTC) [Ballé et al., 2021] is currently the predominant paradigm of neural lossy compression. Compared to transform coding with linear transforms (discussed in Section 2.3.3), non-linear transforms are more flexible and can better adapt to the data distribution [Ballé et al., 2021], and are demonstrated to be optimal in some cases [Wagner and Ballé, 2021]. We convey the main ideas of nonlinear transform coding below, and refer readers to the survey [Ballé et al., 2021] for a more comprehensive discussion.

Nonlinear transform coding replaces the various components of traditional transform coding (Section 2.3.3), such as the analysis transform f, synthesis transform g, and entropy model P, with neural networks or other function approximators, and learn them end-to-end on the data of interest. As in transform coding, we obtain a discrete representation $\hat{\mathbf{z}}$ by quantizing the transform coefficients $\mathbf{z}$, i.e., $\hat{\mathbf{z}} = [\![\mathbf{z}]\!], \mathbf{z} = f(\mathbf{x})$. It is common to refer to $\hat{\mathbf{z}}$ as a tensor of "latents", due to connections to latent variable models (see Section 3.2.1), although there is some ambiguity in this terminology and some papers also apply it to the continuous transform coefficients $\mathbf{z}$. As before, an entropy model P is used to losslessly transmit $\hat{\mathbf{z}}$ via bit-strings (Section 2.2). The entropy models are often based on powerful models from neural lossless compression (Section 3.1), and take the form of a *discretized density model* (discussed in Section 3.1.1),

$$P(\hat{\mathbf{z}}) := \int_{[-0.5,0.5)^n} p(\hat{\mathbf{z}} + \mathbf{v}) d\mathbf{v}, \quad \forall \hat{\mathbf{z}} \in \mathbb{Z}^n. \tag{3.13}$$

Most commonly, the objective in nonlinear transform coding is to simultaneously minimize the rate,

$$\mathcal{R} := \mathbb{E}[-\log_2 P([\![f(\mathbf{X})]\!])], \tag{3.14}$$

and the distortion,

$$\mathcal{D} := \mathbb{E}[\rho(\mathbf{X}, g(\llbracket f(\mathbf{X}) \rrbracket))].$$

The expectations are taken w.r.t. the data distribution $P_{\mathbf{X}}$ and are estimated with data samples. Unlike in Section 2.3, here we no longer concern ourselves with an entropy code γ. Instead, we directly optimize the information content in place of a code length in Eq. 3.14, knowing that an entropy code can (in principle) always be derived from P such that $-\log_2 P(\cdot) \approx |\gamma(\cdot)|$ (see Section 2.2). If we denote the true marginal distribution of $\hat{\mathbf{Z}}$ by $P_{\hat{\mathbf{Z}}}$ (which is induced by $P_{\mathbf{X}}$ and the encoding procedure, via $\hat{\mathbf{Z}} = \llbracket f(\mathbf{X}) \rrbracket$), then the rate loss can be equivalently written as the cross entropy,

$$\mathcal{R} = \mathbb{E}_{\hat{\mathbf{z}} \sim P_{\hat{\mathbf{Z}}}}[-\log_2 P(\hat{\mathbf{z}})], \tag{3.15}$$

which precisely captures the cost of entropy coding $\hat{\mathbf{z}} \sim P_{\hat{\mathbf{Z}}}$ under our model $P(\hat{\mathbf{z}})$, and serves as an upper bound to the entropy $H[\hat{\mathbf{Z}}]$ (see Section 2.2.2).

End-to-end Optimization

As in Eq. 2.12, we would like to optimize the operational R-D performance in the form of a rate-distortion Lagrangian,

$$\mathbb{E}[-\log_2 P(\llbracket f(\mathbf{X}) \rrbracket)] + \lambda \mathbb{E}[\rho(\mathbf{X}, g(\llbracket f(\mathbf{X}) \rrbracket))], \tag{3.16}$$

for a suitable choice of the trade-off factor λ. However, as is written, the objective is not suitable for end-to-end training with SGD, due to the non-differentiability of the quantization operator and the rate loss.

Motivated by the uniform noise approximation to quantization error [Schuchman, 1964],

Ballé et al. [2017] introduced the now popular workaround known as *additive uniform noise* approximation. With this approach, we replace rounding by adding an independent uniform noise,

$$\lfloor \mathbf{z} \rceil \approx \mathbf{z} + \mathbf{u}, \ \mathbf{u} \sim \mathcal{U}\left([-0.5, 0.5)^n\right).$$

Naively, one might simply substitute the above into the Lagrangian Eq. 3.16 and hope to obtain a reasonable training objective. However, the resulting code length, $-\log_2 P(f(\mathbf{x}) + \mathbf{u})$, does not yet make sense, as our entropy model P has only been defined over integers. It turns out the form of P (Eq. 3.13) offers a convenient solution: we can simply extend P from $\mathbb{Z}^n$ to all of $\mathbb{R}^n$, by convolving the underlying density p with the uniform noise $\mathbf{u}$, i.e.,

$$\tilde{p} := p \star \mathcal{U}\left([-0.5, 0.5)^n\right). \tag{3.17}$$

It's easy to see that $\tilde{p}$ agrees with P on all integer points, and serves as a smoothed relaxation of P which defines a surrogate gradient with respect to its input. Replacing P by $\tilde{p}$ in Eq. 3.16, and taking expectation with respect to the uniform noise $\mathbf{u}$, we obtain the surrogate training objective,

$$\mathbb{E}_{\mathbf{x} \sim P_{\mathbf{X}}, \mathbf{u} \sim \mathcal{U}}[-\log_2 \tilde{p}(f(\mathbf{x}) + \mathbf{u}) + \lambda \rho(\mathbf{x}, g(f(\mathbf{x}) + \mathbf{u}))], \tag{3.18}$$

which is now differentiable with respect to all components of the model, and can be simply estimated by Monte-Carlo sampling.

Connection to Variational Autoencoders

An influential idea is to view nonlinear transform coding as equivalently defining a VAE of the data [Ballé et al., 2016, Theis et al., 2017a]. To derive this, it is instructive to consider the

density model p as approximating the distribution of the *continuous* representation $\mathbf{z} = f(\mathbf{x})$ as induced by $\mathbf{x} \sim P_{\mathbf{X}}$ and the analysis transform f. Indeed, suppose $\mathbf{z}$ is distributed according to p (if our modeling is perfect), then the distribution of $\hat{\mathbf{z}} = \lfloor \mathbf{z} \rceil$ has precisely the form of the entropy model P defined by Eq. 3.13. Moreover, if we define the "noisy quantization" by the random variable $\tilde{\mathbf{z}} := \mathbf{z} + \mathbf{u}$, then its induced density (given $\mathbf{z} \sim p$) is precisely $\tilde{p}$ from Eq. 3.17. Based on these connections, the surrogate Lagrangian from additive uniform noise (Eq. 3.18) defines a particular rate-distortion VAE, where $\tilde{\mathbf{z}}$ is the latent variable. Specifically, Eq. 3.18 can be shown [Ballé et al., 2016, Theis et al., 2017a] to be equal to the NELBO objective of an R-D VAE, given by

$$\mathbb{E}_{\mathbf{x}\sim P_{\mathbf{X}}} \mathbb{E}_{\tilde{\mathbf{z}}\sim q(\tilde{\mathbf{z}}|\mathbf{x})}[-\log_2 \tilde{p}(\tilde{\mathbf{z}}) + \log_2 q(\tilde{\mathbf{z}}|\mathbf{x}) - \log_2 p(\mathbf{x}|\tilde{\mathbf{z}})] + \text{const}, \tag{3.19}$$

where $\tilde{p}(\tilde{\mathbf{z}})$ plays the role of a prior, $q(\tilde{\mathbf{z}}|\mathbf{x})$ is a fully factorized uniform posterior density centered at $\mathbf{z} = f(\mathbf{x})$, and $p(\mathbf{x}|\tilde{\mathbf{z}})$ is a likelihood density aligned with the distortion function ρ [5]. The equivalence can be seen by noting that the posterior entropy term is constant (in fact, 0), and sampling from the uniform posterior q is equivalent to adding noise to $f(\mathbf{x})$, by the reparameterization trick.

3.2.2 Channel Simulation and Relative Entropy Coding

We saw that the rate-distortion cost of nonlinear transform coding corresponds to the NELBO objective of a particular latent variable model of the source. In particular, the latent variable model is a variational autoencoder whose inference network computes the location of a box-shaped variational posterior distribution, and the (discretized) prior distribution is used for entropy coding. One may naturally wonder if such a connection between lossy compression and latent variable modeling holds more generally, e.g., without the restriction of a box-shaped

[5] In the common case of a squared distortion, $p(\mathbf{x}|\tilde{\mathbf{z}})$ is a Gaussian with mean equal to the reconstruction $\hat{\mathbf{x}} = g(\tilde{\mathbf{z}})$, and covariance inversely proportional to λ.

variational posterior. In *lossless* compression, the answer is given affirmatively by *bits-back* coding, which achieves (asymptotically) the compression cost given by the NELBO. However, bits-back coding requires the exact same data (hence, variational posterior distribution) to ultimately be available to the receiver, and is therefore only directly applicable to lossless compression.

Reverse channel coding or *channel simulation* [Theis and Yosri, 2021], and closely related, *relative entropy coding* (REC) [Flamich et al., 2020a], serve as a basic bridge between lossy compression and latent variable modeling. A REC algorithm aims to solve the following basic communication problem. Suppose the sender and receiver are each equipped with a common "prior" distribution p and a pseudo-random number generator with a shared, pre-established random seed; the sender (but not receiver) has access to a target distribution q. A REC algorithm allows the sender to transmit a sample $\mathbf{z} \sim q$ using close to $D_{\mathrm{KL}}(q \parallel p)$ bits on average. If q arises from a conditional distribution, e.g., $q_{\mathbf{x}} = q(\mathbf{z} \mid \mathbf{x})$ in the inference distribution of a VAE (which can be viewed as a noisy *channel*), a *reverse channel coding* algorithm allows the sender to transmit $\mathbf{z} \sim q_{\mathbf{x}}$ with $\mathbf{x} \sim p(\mathbf{x})$ using close to $\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[D_{\mathrm{KL}}(q(\mathbf{z} \mid \mathbf{x}) \parallel p(\mathbf{z}))]$ bits on average.

At a high level, a general-purpose REC method works as follows. The sender generates a (possibly large) number of candidate $\mathbf{z}$ samples from the prior p,

$$\mathbf{z}_n \sim p, \quad n = 1, 2, 3, ...,$$

and appropriately chooses an index K such that $\mathbf{z}_K$ is a fair sample from the target distribution, i.e., $\mathbf{z}_K \sim q$. The chosen index $K \in \mathbb{N}$ is then converted to binary and transmitted to the receiver. The receiver recovers $\mathbf{z}_K$ by drawing the same sequence of $\mathbf{z}$ candidates from p (made possible by using a pseudo-random number generator with the same seed as the sender) and stopping at the Kth one.

A major challenge of REC algorithms is that, without restricting the forms of p or q, the computational complexity generally scales exponentially with the amount of information being communicated [Agustsson and Theis, 2020, Goc and Flamich, 2024]. As an example, the MRC algorithm [Cuff, 2008a, Havasi et al., 2018] draws M candidate samples and selects $K \in \{1, 2, , ..., M\}$ with a probability proportional to the importance weights, ${}^{q(\mathbf{z}_n)}/_{p(\mathbf{z}_n)}, n = 1, ..., M$; similarly to in importance sampling [Chatterjee and Diaconis, 2018], M needs to be on the order of $2^{\mathrm{KL}(q\|p)}$ for $\mathbf{z}_K$ to be (approximately) a fair sample from q, thus requiring a number of drawn samples that scales exponentially with the relative entropy $\mathrm{KL}(q\|p)$ (and a coding cost of $\log M \approx \mathrm{KL}(q\|p)$ bits). The exponential complexity prevents, e.g., naively communicating the entire latent tensor $\mathbf{z}$ in a Gaussian VAE for lossy compression, as the relative entropy $D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}))$ easily exceeds thousands of bits, even for a small image. This difficulty can be partly remedied by performing REC on sub-problems with lower dimensions [Flamich et al., 2020a, 2022] for which computationally viable REC algorithms exist [Flamich et al., 2024, Flamich, 2024], but at the expense of worse bit-rate efficiency due to the accumulation of codelength overhead across the dimensions.

Universal Quantization

Here we review a reverse channel coding algorithm for the uniform noise channel, *universal quantization* [Roberts, 1962, Zamir and Feder, 1992], and discuss its connection to nonlinear transform coding. We will see that the differentiable rate cost optimized by nonlinear transform coding during training can be exactly operationalized by applying universal quantization to the latent coefficients [Agustsson and Theis, 2020], instead of deterministic quantization at test time. Thus, we see that universal quantization operationalizes the coding cost of the class of latent variable models associated with nonlinear transform coding (discussed at the end of Section 3.2.1). For more general kinds of latent variables, such as Gaussian VAEs, such a connection is given (in principle) by general-purpose relative entropy coding,

whose computation complexity unfortunately remains intractable and is a subject of ongoing research [Goc and Flamich, 2024].

Suppose the sender has access to a scalar r.v. $Z \sim P_Z$,[6] and would like to communicate a noise-perturbed version of it,

$$\tilde{Z} = Z + U,$$

where $U \sim \mathcal{U}(-0.5, 0.5)$ is an independent r.v. with a uniform distribution on the interval $[-0.5, 0.5]$. In other words, upon observing $Z = z$, the sender aims to transmit a sample from the target distribution $P_{\tilde{Z}|Z=z} = \mathcal{U}(z - 0.5, z + 0.5)$. Universal quantization accomplishes this task as follows:

1. Perturb Z by another independent noise $U' \sim \mathcal{U}(-0.5, 0.5)$, and compute the quantization index $K := \lfloor Z + U' \rceil$, where $\lfloor \cdot \rceil$ denotes rounding to the nearest integer.
2. Entropy code and transmit K under the conditional distribution of K given U'.
3. The receiver draws the same U' by using the same random number generator and obtains a reconstruction $\hat{Z} := K - U' = \lfloor Z + U' \rceil - U'$.

Zamir and Feder [1992] showed that $\hat{Z}$ indeed has the same distribution as $\tilde{Z}$, and remarkably the entropy coding cost of K achieves the ideal rate given by the mutual information between Z and $\tilde{Z}$,

$$H[K|U'] = I(Z; \tilde{Z}) = h(\tilde{Z}) - \cancelto{0}{h(\tilde{Z}|Z)} = h(\tilde{Z}).$$

In the above, $h(\cdot)$ and $h(\cdot|\cdot)$ denote differential entropy and conditional differential entropy coding. In step 2, the (optimal) entropy coding distribution $P_{K|U'=u'}$ can be obtained by

[6] In the context of neural lossy compression, Z would correspond to a scalar coordinate of the encoding transform output $f(\mathbf{X})$.

discretizing the density of $P_{\tilde{Z}}$ on an integer grid offset by $U' = u'$ [Zamir and Feder, 1992],

$$P_{K|U'=u'}(k) = \mathbb{P}(Z+u' \in [k-0.5, k+0.5]) = P_Z([k-u'-0.5, k-u'+0.5]) = f_{\tilde{Z}}(k-u'), \quad k \in \mathbb{N},$$

where $f_{\tilde{Z}}$ denotes the density of $P_{\tilde{Z}}$ given by the convolution $P_{\tilde{Z}} = P_Z \star \mathcal{U}(-0.5, 0.5)$. In practice, the true density $f_{\tilde{Z}}$ is unknown, so we replace it with a surrogate density model $f_\theta(\tilde{z})$ and incur a higher coding cost, given by an expected relative entropy

$$\mathbb{E}_{z\sim P_Z}[D_{\mathrm{KL}}(f_{\tilde{Z}|Z=z}(\cdot) \parallel f_\theta(\cdot))] \geq I(Z;\tilde{Z}), \tag{3.20}$$

where $f_{\tilde{Z}|Z=z} = \mathbb{1}_{[z-0.5,z+0.5]}$ denotes the density function of the uniform noise channel $P_{\tilde{Z}|Z=z}$.[7] The LHS is the common variational upper bound on mutual information [Polyanskiy and Wu, 2022, Theorem 4.1],

$$\mathbb{E}_{z\sim P_Z}[D_{\mathrm{KL}}(f_{\tilde{Z}|Z=z}(\cdot) \parallel f_\theta(\cdot))] = I(Z;\tilde{Z}) + D_{\mathrm{KL}}(f_{\tilde{Z}} \parallel f_\theta),$$

and the bound is tight when $f_{\tilde{Z}} = f_\theta$. A popular modeling choice for f_θ is the noise-convolution of an underlying density model $p_\theta(z)$, as in Eq. 3.17 earlier, $f_\theta = p_\theta \star \mathcal{U}(-0.5, 0.5)$. [8]

Rewriting the LHS further, we see that it is precisely equal to the differentiable rate term being optimized in nonlinear transform coding,

$$\begin{aligned}\mathbb{E}_{z\sim P_Z}[D_{\mathrm{KL}}(f_{\tilde{Z}|Z=z}(\cdot) \parallel f_\theta(\cdot))] &= h[f_{\tilde{Z}}, f_\theta] - \cancelto{0}{h(\tilde{Z}|Z)} \\ &= \mathbb{E}_{z\sim P_Z, u\sim\mathcal{U}, \tilde{z}=z+u}[-\log f_\theta(\tilde{z})],\end{aligned}$$

and this cost is operationalized by universal quantization with an entropy model derived from

[7] To match the earlier setup of Section 3.2.2, we have a family of relative entropy coding problems for each $Z = z$, where the target distribution is given by $P_{\tilde{Z}|Z=z}$, and f_θ is the shared prior. The ideal coding cost is now an *expectation* of relative entropy w.r.t. $Z \sim P_Z$.

[8] Suppose Z has density f_Z; then $D_{\mathrm{KL}}(f_{\tilde{Z}} \parallel f_\theta) = D_{\mathrm{KL}}(f_Z \star \mathcal{U}(-0.5, 0.5) \parallel p_\theta \star \mathcal{U}(-0.5, 0.5))$ is a form of *spread divergence* [Zhang et al., 2020] between f_Z and p_θ.

p_θ, i.e.,

$$\mathbb{E}_{z\sim P_Z}[D_{\mathrm{KL}}(f_{\tilde{Z}|Z=z}(\cdot)\parallel f_\theta(\cdot))] = \mathbb{E}_{z\sim P_Z,u'\sim\mathcal{U}}[H[P_{K|U'=u'}, P_{\theta,u'}]]$$

where $H[\cdot,\cdot]$ denotes cross-entropy, and

$$P_{\theta,u'}(k) := f_\theta(k-u') = \int_{k-u'-0.5}^{k-u'+0.5} p_\theta(t)dt, \quad k\in\mathbb{N}$$

is the entropy model derived from p_θ.

Chapter 4

Inference Optimization

Most of the material in this chapter has been published in the following conference paper,

Improving Inference for Neural Image Compression. Yibo Yang, Robert Bamler, and Stephan Mandt. *Conference on Neural Information Processing Systems (NeurIPS), 2020.*

Additional results and supplementary information can be found in Appendix A, and the project repo can be found at https://github.com/mandt-lab/improving-inference-for-neural-image-compression.

A word on notation: in this chapter we follow the probabilistic machine learning convention, where $\mathbf{y}$ is a vector of latent variables with a variational posterior distribution $q(\mathbf{y}|\mathbf{x})$ parameterized by mean $\mu_{\mathbf{y}}$ and variance $\sigma^2_{\mathbf{y}}$. The mean parameters $\mu_{\mathbf{y}}$ correspond to the (pre-quantization) transform coefficients in nonlinear transform coding. Similarly, $\mathbf{z}$ denotes a vector of hyper latent variables with a variational posterior distribution $q(\mathbf{z}|\mathbf{x})$ parameterized by mean $\mu_{\mathbf{z}}$ and variance $\sigma^2_{\mathbf{z}}$.

We consider the problem of lossy image compression with deep latent variable models. State-of-the-art methods [Ballé et al., 2018b, Minnen et al., 2018, Lee et al., 2019a] build on hierarchical variational autoencoders (VAEs) and learn inference networks to predict a compressible latent representation of each data point. Drawing on the variational inference perspective on compression [Alemi et al., 2018], we identify three approximation gaps which limit performance in the conventional approach: an amortization gap, a discretization gap, and a marginalization gap. We propose remedies for each of these three limitations based on ideas related to iterative inference, stochastic annealing for discrete optimization, and bits-back coding, resulting in the first application of bits-back coding to lossy compression. In our experiments, which include extensive baseline comparisons and ablation studies, we achieve new state-of-the-art performance on lossy image compression using an established VAE architecture, by changing only the inference method.

4.1 Introduction

Deep learning methods are reshaping the field of data compression, and recently started to outperform state-of-the-art classical codecs on image compression [Minnen et al., 2018]. Besides useful on its own, image compression is a stepping stone towards better video codecs [Han et al., 2019, Habibian et al., 2019, Yang et al., 2021], which can reduce a sizable amount of global internet traffic.

State-of-the-art neural methods for lossy image compression [Ballé et al., 2018b, Minnen et al., 2018, Lee et al., 2019a] learn a mapping between images and latent variables with a variational autoencoder (VAE). An *inference network* maps a given image to a compressible latent representation, which a generative network can then map back to a reconstructed image. In fact, compression can be more broadly seen as a form of inference: to compress—or "encode"—data, one has to perform inference over a well-specified decompression—or

"decoding"—algorithm.

In classical compression codecs, the *de*coder has to follow a well-specified procedure to ensure interoperability between different implementations of the same codec. By contrast, the *en*coding process is typically not uniquely defined: different encoder implementations of the same codec often compress the same input data to different bitstrings. For example, `pngcrush` [Randers-Pehrson, 1997] typically produces smaller PNG files than `ImageMagick`. Yet, both programs are standard-compliant PNG encoder implementations as both produce a compressed file that decodes to the same image. In other words, both encoder implementations perform *correct* inference over the standardized decoder specification, but use different inference algorithms with different performance characteristics.

The insight that better inference leads to better compression performance even within the same codec motivates us to reconsider how inference is typically done in neural data compression with VAEs. In this paper, we show that the conventional *amortized* inference [Kingma and Welling, 2014, Rezende et al., 2014a] in VAEs leaves substantial room for improvement when used for data compression.

We propose an improved inference method for data compression tasks with three main innovations:

1. *Improved amortization:* The amortized inference strategy in VAEs speeds up training, but is restrictive at compression time. We draw a connection between a recently proposed iterative procedure for compression [Campos et al., 2019] to the broader literature of VI that closes the amortization gap, which provides the basis of the following two novel inference methods.

2. *Improved discretization:* Compression requires discretizing the latent representation from VAEs, because only discrete values can be entropy coded. As inference over discrete variables is difficult, existing methods typically relax the discretization constraint in

some way during inference and then discretize afterwards. We instead propose a novel method based on a stochastic annealing scheme that performs inference directly over discrete points.

3. *Improved entropy coding:* In lossless compression with latent-variable models, bits-back coding [Wallace, 1990, Hinton and Van Camp, 1993] allows approximately coding the latents with the *marginal* prior. It is so far believed that bits-back coding is incompatible with lossy compression [Habibian et al., 2019] because it requires inference on the *decoder* side, which does not have access to the exact (undistorted) input data. We propose a remedy to this limitation, resulting in the first application of bits-back coding to lossy compression.

We evaluate the above three innovations on an otherwise unchanged architecture of an established model and compare against a wide range of baselines and ablations. Our proposals significantly improve compression performance and results in a new state of the art in lossy image compression.

The rest of the paper is structured as follows: Section 4.2 summarizes lossy compression with VAEs, for which Section 4.3 proposes the above three improvements. Section 4.4 reports experimental results. We conclude in Section 4.5. We review related work in each relevant subsection.

4.2 Background: Lossy Neural Image Compression as Variational Inference

In this section, we summarize an existing framework for lossy image compression with deep latent variable models, which will be the basis of three proposed improvements in Section 4.3.

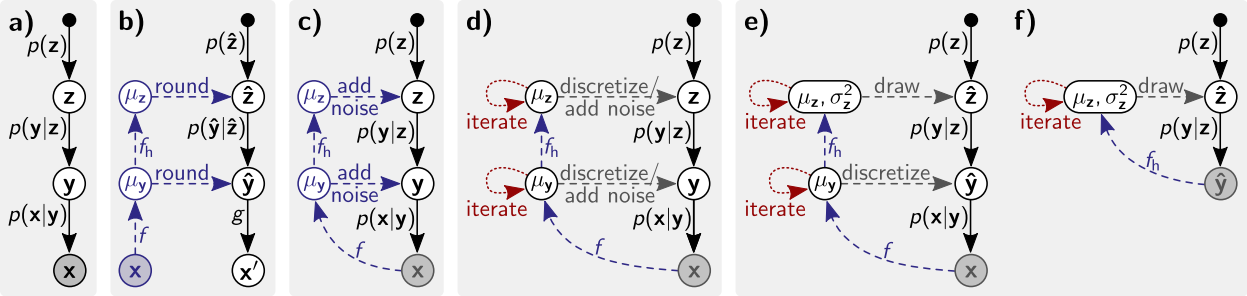


Figure 4.1: Graphical model and control flow charts. a) generative model with hyperlatents $\mathbf{z}$, latents $\mathbf{y}$, and image $\mathbf{x}$ [Minnen et al., 2018]; b) conventional method for compression (dashed blue, see Eq. 4.1) and decompression (solid black); c) common training objective due to [Ballé et al., 2017] (see Eq. 4.3); d) proposed hybrid amortized (dashed blue) / iterative (dotted red) inference (Section 4.3.1); e)-f) inference in the proposed lossy bitsback method (Section 4.3.3); the encoder first executes e) and then keeps $\hat{\mathbf{y}}$ fixed while executing f); the decoder reconstructs $\hat{\mathbf{y}}$ and then executes f) to get bits back.

Related Work. Ballé et al. [2017] and Theis et al. [2017b] were among the first to recognize a connection between the rate-distortion objective of lossy compression and the loss function of a certain kind of Variational Autoencoders (VAEs), and to apply it to end-to-end image compression. [Ballé et al., 2018b] proposes a hierarchical model, upon which current state-of-the-art methods [Minnen et al., 2018, Lee et al., 2019a] further improve by adding an auto-regressive component. For simplicity, we adopt the VAE of [Minnen et al., 2018] without the auto-regressive component, reviewed in this section below, which has also been a basis for recent compression research [Johnston et al., 2019].

Generative Model. Figure 4.1a shows the generative process of an image $\mathbf{x}$. The Gaussian likelihood $p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}; g(\mathbf{y}; \theta), \sigma_{\mathbf{x}}^2 I)$ has a fixed variance $\sigma_{\mathbf{x}}^2$ and its mean is computed from latent variables $\mathbf{y}$ by a deconvolutional neural network (DNN) g with weights θ. A second DNN g_{h} with weights θ_{h} outputs the parameters (location and scale) of the prior $p(\mathbf{y}|\mathbf{z})$ conditioned on "hyperlatents" $\mathbf{z}$.

Compression and Decompression. Figure 4.1b illustrates compression ("encoding", dashed blue) and decompression ("decoding", solid black) as proposed in [Minnen et al.,

2018]. The encoder passes a target image $\mathbf{x}$ through trained inference networks f and f_{h} with weights ϕ and ϕ_{h}, respectively. The resulting continuous latent representations, $(\mu_{\mathbf{y}}, \mu_{\mathbf{z}})$, are rounded (denoted $\lfloor \cdot \rceil$) to discrete $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$,

$$\hat{\mathbf{y}} = \lfloor \mu_{\mathbf{y}} \rceil; \quad \hat{\mathbf{z}} = \lfloor \mu_{\mathbf{z}} \rceil \qquad \text{where} \qquad \mu_{\mathbf{y}} = f(\mathbf{x}; \phi); \quad \mu_{\mathbf{z}} = f_{\mathrm{h}}(f(\mathbf{x}; \phi); \phi_{\mathrm{h}}). \tag{4.1}$$

The encoder then entropy-codes $\hat{\mathbf{z}}$ and $\hat{\mathbf{y}}$, using discretized versions $P(\hat{\mathbf{z}})$ and $P(\hat{\mathbf{y}}|\hat{\mathbf{z}})$ of the hyperprior $p(\mathbf{z})$ and (conditional) prior $p(\mathbf{y}|\mathbf{z})$, respectively, as entropy models. This step is simplified by a clever restriction on the shape of p, such that it agrees with the discretized P on all integers (see [Ballé et al., 2017, 2018b, Minnen et al., 2018]). The decoder then recovers $\hat{\mathbf{z}}$ and $\hat{\mathbf{y}}$ and obtains a lossy image reconstruction $\hat{\mathbf{x}} := \arg\max_{\mathbf{x}} p(\mathbf{x}|\hat{\mathbf{y}}) = g(\hat{\mathbf{y}}; \theta)$.

Model Training. Let $\mathbf{x}$ be sampled from a training set of images. To train the above VAE for lossy compression, Theis et al. [2017b], Minnen et al. [2018] consider minimizing a rate-distortion objective:

$$\begin{aligned} \mathcal{L}_\lambda(\lfloor \mu_{\mathbf{y}} \rceil, \lfloor \mu_{\mathbf{z}} \rceil) &= \mathcal{R}(\lfloor \mu_{\mathbf{y}} \rceil, \lfloor \mu_{\mathbf{z}} \rceil) + \lambda \mathcal{D}(\lfloor \mu_{\mathbf{y}} \rceil, \mathbf{x}) \\ &= \underbrace{-\log_2 p\big(\lfloor \mu_{\mathbf{z}} \rceil\big)}_{\substack{\text{information content} \\ \text{of } \hat{\mathbf{z}} = \lfloor \mu_{\mathbf{z}} \rceil}} \underbrace{-\log_2 p\big(\lfloor \mu_{\mathbf{y}} \rceil \,\big|\, \lfloor \mu_{\mathbf{z}} \rceil\big)}_{\substack{\text{information content of} \\ \hat{\mathbf{y}} = \lfloor \mu_{\mathbf{y}} \rceil \text{ given } \hat{\mathbf{z}} = \lfloor \mu_{\mathbf{z}} \rceil}} + \lambda \underbrace{\big|\big|\mathbf{x} - g\big(\lfloor \mu_{\mathbf{y}} \rceil; \theta\big)\big|\big|_2^2}_{\text{distortion } \mathcal{D}} \end{aligned} \tag{4.2}$$

where $(\mu_{\mathbf{y}}, \mu_{\mathbf{z}})$ is computed from the inference networks as in Eq. 4.1, and the parameter $\lambda > 0$ controls the trade-off between bitrate $\mathcal{R}$ under entropy coding, and distortion (reconstruction error) $\mathcal{D}$. As the rounding operations prevent gradient-based optimization, Ballé et al. [2017] propose to replace rounding during training by adding uniform noise from the interval $[-\frac{1}{2}, \frac{1}{2}]$ to each coordinate of $\mu_{\mathbf{y}}$ and $\mu_{\mathbf{z}}$ (see Figure 4.1c). This is equivalent to sampling from uniform distributions $q(\mathbf{y}|\mathbf{x})$ and $q(\mathbf{z}|\mathbf{x})$ with a fixed width of one centered around $\mu_{\mathbf{y}}$ and $\mu_{\mathbf{z}}$, respectively. One thus obtains the following relaxed rate-distortion objective, for a given

data point $\mathbf{x}$,

$$\tilde{\mathcal{L}}_\lambda(\theta, \theta_\mathrm{h}, \phi, \phi_\mathrm{h}) = \mathbb{E}_{q(\mathbf{y}|\mathbf{x})\, q(\mathbf{z}|\mathbf{x})}\left[-\log_2 p(\mathbf{z}) - \log_2 p(\mathbf{y}|\mathbf{z}) + \lambda||\mathbf{x} - g(\mathbf{y};\theta)||_2^2\right]. \tag{4.3}$$

Connection to Variational Inference. As pointed out in [Ballé et al., 2017], the relaxed objective in Eq. 4.3 is the negative evidence lower bound (NELBO) of variational inference (VI) if we identify $\lambda = 1/(2\sigma_\mathbf{x}^2 \log 2)$. This draws a connection between lossy compression and VI [Blei et al., 2017, Zhang et al., 2018]. We emphasize a distinction between variational *inference* and variational *expectation maximization* (EM) [Beal and Ghahramani, 2003a]: whereas variational EM trains a model (and employs VI as a subroutine), VI is used to compress data using a trained model. A central result of this paper is that improving inference in a fixed generative model at compression time already suffices to significantly improve compression performance.

4.3 Novel Inference Techniques for Data Compression

This section presents our main contributions. We identify three approximation gaps in VAE-based compression methods (see Section 4.2): an amortization gap, a discretization gap, and a marginalization gap. Recently, the amortization gap was considered by [Campos et al., 2019]; we expand on this idea in Section 4.3.1, bringing it under a wider framework of algorithms in the VI literature. We then propose two specific methods that improve inference at compression time: a novel inference method over discrete representations (Section 4.3.2) that closes the discretization gap, and a novel lossy bits-back coding method (Section 4.3.3) that closes the marginalization gap.

4.3.1 Amortization Gap and Hybrid Amortized-Iterative Inference

Amortization Gap. Amortized variational inference [Kingma and Welling, 2013, Rezende et al., 2014b] turns optimization over *local* (per-data) variational parameters into optimization over the *global* weights of an inference network. In the VAE of Section 4.2, the inference networks f and f_{h} map an image $\mathbf{x}$ to local parameters $\mu_{\mathbf{y}}$ and $\mu_{\mathbf{z}}$ of the variational distributions $q(\mathbf{y}|\mathbf{x})$ and $q(\mathbf{z}|\mathbf{x})$, respectively (Eq. 4.1). Amortized VI speeds up training by avoiding an expensive inner inference loop of variational EM, but it leads to an amortization gap [Cremer et al., 2018, Krishnan et al., 2018], which is the difference between the value of the NELBO (Eq. 4.3) when $\mu_{\mathbf{y}}$ and $\mu_{\mathbf{z}}$ are obtained from the inference networks, compared to its true minimum when $\mu_{\mathbf{y}}$ and $\mu_{\mathbf{z}}$ are directly minimized. Since the NELBO approximates the rate-distortion objective (Section 4.2), the amortization gap translates into sub-optimal performance when amortization is used at compression time. As discussed below, this gap can be closed by refining the output of the inference networks by iterative inference.

Related Work. The idea of combining amortized inference with iterative optimization has been studied and refined by various authors [Hjelm et al., 2016, Kim et al., 2018, Krishnan et al., 2018, Marino et al., 2018]. While not formulated in the language of variational autoencoders and variational inference, Campos et al. [2019] apply a simple version of this idea to compression. Drawing on the connection to hybrid amortized-iterative inference, we show that the method can be drastically improved by addressing the discretization gap (Section 4.3.2) and the marginalization gap (Section 4.3.3).

Hybrid Amortized-Iterative Inference. We reinterpret the proposal in [Campos et al., 2019] as a basic version of a hybrid amortized-iterative inference idea that changes inference only at compression time but not during model training. Figure 4.1d illustrates the approach. When compressing a target image $\mathbf{x}$, one initializes $\mu_{\mathbf{y}}$ and $\mu_{\mathbf{z}}$ from the trained inference

networks f and f_{h}, see Eq. 4.1. One then treats $\mu_{\mathbf{y}}$ and $\mu_{\mathbf{z}}$ as *local* variational parameters, and minimizes the NELBO (Eq. 4.3) over $(\mu_{\mathbf{y}}, \mu_{\mathbf{z}})$ with the reparameterization trick and stochastic gradient descent. This approach thus separates inference *at test time* from inference *during model training*. We show in the next two sections that this simple idea forms a powerful basis for new inference approaches that drastically improve compression performance.

4.3.2 Discretization Gap and Stochastic Gumbel Annealing (SGA)

Discretization Gap. Compressing data to a bitstring is an inherently *discrete* optimization problem. As discrete optimization in high dimensions is difficult, neural compression methods instead optimize some relaxed objective function (such as the NELBO $\tilde{\mathcal{L}}_\lambda$ in Eq. 4.3) over continuous representations (such as $\mu_{\mathbf{y}}$ and $\mu_{\mathbf{z}}$), which are then discretized afterwards for entropy coding (see Eq. 4.1). This leads to a *discretization gap*: the difference between the true rate-distortion objective $\mathcal{L}_\lambda$ at the discretized representation $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$, and the relaxed objective function $\tilde{\mathcal{L}}_\lambda$ at the continuous approximation $(\mu_{\mathbf{y}}, \mu_{\mathbf{z}})$.

Related Work. Current neural compression methods use a differentiable approximation to discretization *during model training*, such as Straight-Through Estimator (STE) [Bengio et al., 2013, Oord et al., 2017, Yin et al., 2019], adding uniform noise [Ballé et al., 2017] (see Eq. 4.3), stochastic binarization [Toderici et al., 2016], and soft-to-hard quantization [Agustsson et al., 2017]. Discretization *at compression time* was addressed in [Yang et al., 2020b] without assuming that the training procedure takes discretization into account, whereas our work makes this additional assumption.

Stochastic Gumbel Annealing (SGA). We refine the hybrid amortized-iterative inference approach of Section 4.3.1 to close the discretization gap: for a given image $\mathbf{x}$, we aim to find its *discrete* (in our case, integer) representation $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ that optimizes the rate-distortion

objective $\mathcal{L}_\lambda$ in Eq. 4.2, this time re-interpreted as a function of $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ directly. Our proposed annealing scheme approaches this discretization optimization problem with the help of continuous proxy variables $\mu_{\mathbf{y}}$ and $\mu_{\mathbf{z}}$, which we initialize from the inference networks as in Eq. 4.1. We limit the following discussion to the latents $\mu_{\mathbf{y}}$ and $\hat{\mathbf{y}}$, treating the hyperlatents $\mu_{\mathbf{z}}$ and $\hat{\mathbf{z}}$ analogously. For each dimension i of $\mu_{\mathbf{y}}$, we map the continuous coordinate $\mu_{\mathbf{y},i} \in \mathbb{R}$ to an integer coordinate $\hat{\mathbf{y}}_i \in \mathbb{Z}$ by rounding either up or down. Let $r_{\mathbf{y},i}$ be a one-hot vector that indicates the rounding direction, with $r_{\mathbf{y},i} = (1, 0)$ for rounding down (denoted $\lfloor \cdot \rfloor$) and $r_{\mathbf{y},i} = (0, 1)$ for rounding up (denoted $\lceil \cdot \rceil$). Thus, the result of rounding is the inner product, $\hat{\mathbf{y}}_i = r_{\mathbf{y},i} \cdot (\lfloor \mu_{\mathbf{y},i} \rfloor, \lceil \mu_{\mathbf{y},i} \rceil)$. Now we let $r_{\mathbf{y},i}$ be a Bernoulli variable with a "tempered" distribution

$$q_\tau(r_{\mathbf{y},i} | \mu_{\mathbf{y},i}) \propto \begin{cases} \exp\big\{ -\psi\big(\mu_{\mathbf{y},i} - \lfloor \mu_{\mathbf{y},i} \rfloor\big)/\tau \big\} & \text{if } r_{\mathbf{y},i} = (1, 0) \\ \exp\big\{ -\psi\big(\lceil \mu_{\mathbf{y},i} \rceil - \mu_{\mathbf{y},i}\big)/\tau \big\} & \text{if } r_{\mathbf{y},i} = (0, 1) \end{cases} \tag{4.4}$$

with temperature parameter $\tau > 0$, and $\psi : [0, 1) \to \mathbb{R}$ is an increasing function satisfying $\lim_{\alpha \to 1} \psi(\alpha) = \infty$ (this is chosen to ensure continuity of the resulting objective (4.5); we used $\psi = \tanh^{-1}$). Thus, as $\mu_{\mathbf{y},i}$ approaches an integer, the probability of rounding to that integer approaches one. Defining $q_\tau(r_{\mathbf{y}} | \mu_{\mathbf{y}}) = \prod_i q_\tau(r_{\mathbf{y},i} | \mu_{\mathbf{y},i})$, we thus minimize the stochastic objective

$$\tilde{\mathcal{L}}_{\lambda,\tau}(\mu_{\mathbf{y}}, \mu_{\mathbf{z}}) = \mathbb{E}_{q_\tau(r_{\mathbf{y}} | \mu_{\mathbf{y}})\, q_\tau(r_{\mathbf{z}} | \mu_{\mathbf{z}})} \Big[\mathcal{L}_\lambda \Big(r_{\mathbf{y}} \cdot \big(\lfloor \mu_{\mathbf{y}} \rfloor, \lceil \mu_{\mathbf{y}} \rceil\big), r_{\mathbf{z}} \cdot \big(\lfloor \mu_{\mathbf{z}} \rfloor, \lceil \mu_{\mathbf{z}} \rceil\big) \Big) \Big] \tag{4.5}$$

where we reintroduce the hyperlatents $\mathbf{z}$, and $r_{\mathbf{y}} \cdot (\lfloor \mu_{\mathbf{y}} \rfloor, \lceil \mu_{\mathbf{y}} \rceil)$ denotes the discrete vector $\hat{\mathbf{y}}$ obtained by rounding each coordinate $\mu_{\mathbf{y},i}$ of $\mu_{\mathbf{y}}$ according to its rounding direction $r_{\mathbf{y},i}$. At any temperature τ, the objective in Eq. 4.5 smoothly interpolates the R-D objective in Eq. 4.2 between all integer points. We minimize Eq. 4.5 over $(\mu_{\mathbf{y}}, \mu_{\mathbf{z}})$ with stochastic gradient descent, propagating gradients through Bernoulli samples via the Gumbel-softmax trick [Jang et al., 2016, Maddison et al., 2016] using, for simplicity, the same temperature τ

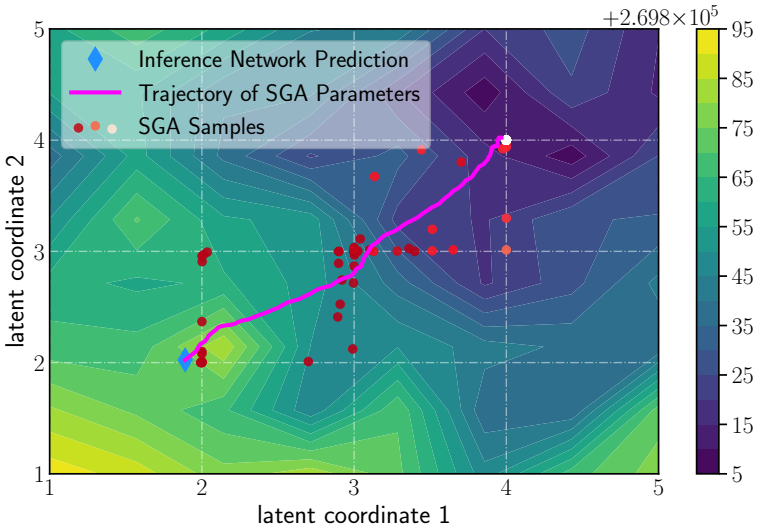


Figure 4.2: Visualizing SGA optimization. White dashed lines indicate the discretization grid. SGA samples are colored by temperature (lighter color corresponds to lower temperature). 10 Gumbel-softmax samples were used to approximate expectations in the objective (4.5), and temperature $\tau = 0.1$.

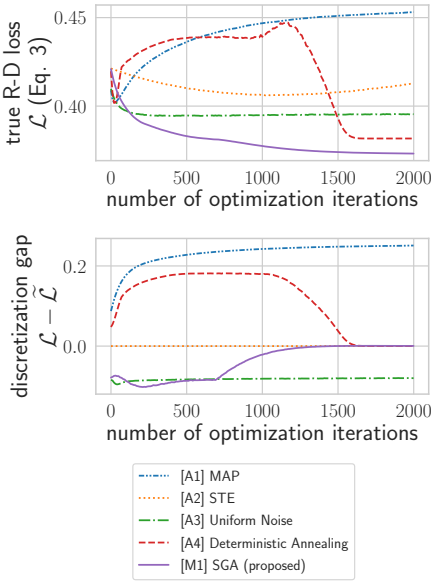


Figure 4.3: Comparing Stochastic Gumbel Annealing (Section 4.3.2) to alternatives (Table 4.1, [A1]-[A4]).

as in q_τ. We note that in principle, REINFORCE [Williams, 1992] can also work instead of Gumbel-softmax. We anneal τ towards zero over the course of optimization, such that the Gumbel approximation becomes exact and the stochastic rounding operation converges to the deterministic one in Eq. 4.1. We thus name this method "Stochastic Gumbel Annealing" (SGA).

Figure 4.2 illustrates the loss function (Eq. 4.5) landscape of SGA near its converged solution, for a given Kodak image. Here, we visualize along the two coordinates of $\mu_{\mathbf{y}}$ whose optimization trajectories have the most variance among all coordinates, plotting the optimization trajectory of $\mu_{\mathbf{y}}$ (magenta line) and the stochastically rounded samples (red dots) from the inner product $r_{\mathbf{y}} \cdot \big(\lfloor \mu_{\mathbf{y}} \rfloor, \lceil \mu_{\mathbf{y}} \rceil\big)$. The final SGA sample converges to the deterministically rounded solution (white dot, $(4, 4)$), which is significantly better than the initialization predicted by inference network (blue diamond, near $(2, 2)$).

Comparison to Alternatives. Alternatively, we could have started with Eq. 4.2, and optimized it as a function of $(\mu_{\mathbf{y}}, \mu_{\mathbf{z}})$, using existing discretization methods to relax rounding; the optimized $(\mu_{\mathbf{y}}, \mu_{\mathbf{z}})$ would subsequently be rounded to $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$. However, our method outperforms all these alternatives. Specifically, we tried (with shorthands [A1]-[A4] for 'ablation', referring to Table 4.1 of Section 4.4): [A1] **MAP**: completely ignoring rounding during optimization; [A2] **Straight-Through Estimator (STE)** [Bengio et al., 2013]: rounding only on the forward pass of automatic differentiation; [A3] **Uniform Noise** [Campos et al., 2019]: optimizing Eq. 4.3 over $(\mu_{\mathbf{y}}, \mu_{\mathbf{z}})$ at compression time, following the Hybrid Amortized-Iterative Inference approach of Section 4.3.1; and [A4] **Deterministic Annealing**: turning the stochastic rounding operations of SGA into deterministic weighted averages, by pushing the expectation operators w.r.t q in Eq. 4.5 to *inside* the arguments of $\mathcal{L}_\lambda$, resulting in the loss function $\mathcal{L}_\lambda\Big(\mathbb{E}_{q_\tau(r_{\mathbf{y}}|\mu_{\mathbf{y}})}[r_{\mathbf{y}} \cdot (\lfloor \mu_{\mathbf{y}} \rfloor, \lceil \mu_{\mathbf{y}} \rceil)], \mathbb{E}_{q_\tau(r_{\mathbf{z}}|\mu_{\mathbf{z}})}[r_{\mathbf{z}} \cdot (\lfloor \mu_{\mathbf{z}} \rfloor, \lceil \mu_{\mathbf{z}} \rceil)]\Big)$ resembling [Agustsson et al., 2017].

We tested the above discretization methods on images from Kodak [Kodak, 1993], using a pre-trained hyperprior model (Section 4.2). Figure 4.3 (left) shows learning curves for the true rate-distortion objective $\mathcal{L}_\lambda$, evaluated via rounding the intermediate $(\mu_{\mathbf{y}}, \mu_{\mathbf{z}})$ throughout optimization. Figure 4.3 (right) shows the discretization gap $\mathcal{L}_\lambda - \tilde{\mathcal{L}}_\lambda$, where $\tilde{\mathcal{L}}_\lambda$ is the relaxed objective of each method. Our proposed *SGA* method closes the discretization gap and achieves the lowest true rate-distortion loss among all methods compared, using the same initialization. *Deterministic Annealing* also closed the discretization gap, but converged to a worse solution. *MAP* naively converged to a non-integer solution, and *STE* consistently diverged even with a tiny learning rate, as also observed by Yin et al. [2019] (following their results, we changed the gradient of the backward pass from identity to that of ReLU or clipped ReLU; neither helped). *Uniform Noise* produced a consistently negative discretization gap, as the noise-perturbed objective in Eq. 4.3 empirically overestimates the true rate-distortion objective $\mathcal{L}_\lambda$ on a large neighborhood around the initial $(\mu_{\mathbf{y}}, \mu_{\mathbf{z}})$ computed by the inference network; by contrast, SGA shrinks this gap by letting its variational distribution q increasingly

concentrate on integer solutions.

4.3.3 Marginalization Gap and Lossy Bits-Back Coding

Marginalization Gap. To compress an image $\mathbf{x}$ with the hierarchical model of Section 4.2, ideally we would only need to encode and transmit its latent representation $\hat{\mathbf{y}}$ but not the hyper-latents $\hat{\mathbf{z}}$, as only $\hat{\mathbf{y}}$ is needed for reconstruction. This would require entropy coding with (a discretization of) the *marginal* prior $p(\mathbf{y}) = \int p(\mathbf{z})\, p(\mathbf{y}|\mathbf{z})\, d\mathbf{z}$, which unfortunately is computationally intractable. Therefore, the standard compression approach reviewed in Section 4.2 instead encodes some hyperlatent $\hat{\mathbf{z}}$ using the discretized hyperprior $P(\hat{\mathbf{z}})$ first, and then encodes $\hat{\mathbf{y}}$ using the entropy model $P(\hat{\mathbf{y}}|\hat{\mathbf{z}})$. This leads to a *marginalization gap*, which is the difference between the information content $-\log_2 P(\hat{\mathbf{z}}, \hat{\mathbf{y}}) = -\log_2 P(\hat{\mathbf{z}}) - \log_2 P(\hat{\mathbf{y}}|\hat{\mathbf{z}})$ of the transmitted tuple $(\hat{\mathbf{z}}, \hat{\mathbf{y}})$ and the information content of $\hat{\mathbf{y}}$ alone,

$$-\log_2 P(\hat{\mathbf{z}}, \hat{\mathbf{y}}) - (-\log_2 P(\hat{\mathbf{y}})) = -\log_2 P(\hat{\mathbf{z}}|\hat{\mathbf{y}}), \tag{4.6}$$

i.e., the marginalization gap is the information content of the hyperlatent $\hat{\mathbf{z}}$ under the ideal posterior $P(\hat{\mathbf{z}}|\hat{\mathbf{y}})$, in the (discrete) latent generative process $\hat{\mathbf{z}} \to \hat{\mathbf{y}}$.

Related Work. A similar marginalization gap has been addressed in *lossless* compression by bits-back coding [Wallace, 1990, Hinton and Van Camp, 1993] and made more practical by the BB-ANS algorithm [Townsend et al., 2019a]. Recent work has improved its efficiency in hierarchical latent variable models [Townsend et al., 2019b, Kingma et al., 2019], and extended it to flow models [Ho et al., 2019a]. To our knowledge, bits-back coding has not yet been used in lossy compression. This is likely since bits-back coding requires Bayesian posterior inference on the *decoder* side, which seems incompatible with lossy compression, as the decoder does not have access to the undistorted data $\mathbf{x}$.

Lossy Bits-Back Coding. We extend the bits-back idea to lossy compression by noting that the discretized latent representation $\hat{\mathbf{y}}$ is encoded losslessly (Figure 4.1b) and thus amenable to bits-back coding. A complication arises as bits-back coding requires that the encoder's inference over $\mathbf{z}$ can be exactly reproduced by the decoder (see below). This requirement is violated by the inference network in Eq. 4.1, where $\mu_{\mathbf{z}} = f_{\mathrm{h}}(f(\mathbf{x};\phi);\phi_{\mathrm{h}})$ depends on $\mathbf{x}$, which is not available to the decoder in lossy compression. It turns out that the naive fix of setting instead $\mu_{\mathbf{z}} = f_{\mathrm{h}}(\hat{\mathbf{y}};\phi_{\mathrm{h}})$ would hurt performance by more than what bits-back saves (see ablations in Section 4.4). We propose instead a two-stage inference algorithm that cuts the dependency on $\mathbf{x}$ after an initial joint inference over $\hat{\mathbf{y}}$ and $\mathbf{z}$.

Bits-back coding bridges the marginalization gap in Eq. 4.6 by encoding a limited number of bits of some additional side information (e.g., an image caption or a previously encoded image of a slide show) into the choice of $\hat{\mathbf{z}}$ using an entropy model $Q(\hat{\mathbf{z}}|\mu_{\mathbf{z}},\sigma^2_{\mathbf{z}})$ with parameters $\mu_{\mathbf{z}}$ and $\sigma^2_{\mathbf{z}}$. We obtain $Q(\hat{\mathbf{z}}|\mu_{\mathbf{z}},\sigma^2_{\mathbf{z}})$ by discretizing a Gaussian variational distribution $q(\mathbf{z}|\mu_{\mathbf{z}},\sigma^2_{\mathbf{z}}) = \mathcal{N}(\mathbf{z};\mu_{\mathbf{z}},\mathrm{diag}(\sigma^2_{\mathbf{z}}))$, thus extending the last layer of the hyper-inference network f_{h} (Eq. 4.1) to output now a tuple $(\mu_{\mathbf{z}},\sigma^2_{\mathbf{z}})$ of means and diagonal variances. This replaces the form of variational distribution $q(\mathbf{z}|\mathbf{x})$ of Eq 4.3 as the restriction to a box-shaped distribution with fixed width is not necessary in bits-back coding. Additionally, we drop the restriction on the shape of the hyperprior $p(\mathbf{z})$ reviewed in Section 4.2, simply adopting the flexible density model as proposed in [Ballé et al., 2018b] without convolving it with a uniform distribution, as $\mu_{\mathbf{z}}$ is no longer discretized to integers. We train the resulting VAE by minimizing the NELBO. Since $q(\mathbf{z}|\mu_{\mathbf{z}},\sigma^2_{\mathbf{z}})$ now has a variable width, the NELBO has an extra term compared to Eq. 4.3 that subtracts the entropy of $q(\mathbf{z}|\mu_{\mathbf{z}},\sigma^2_{\mathbf{z}})$, reflecting the expected number of bits we 'get back'. Thus, the NELBO is again a relaxed rate-distortion objective, where now the *net* rate is the compressed file size minus the amount of embedded side information.

Algorithm 1 describes lossy compression and decompression with the trained model. Subroutine `encode` initializes the variational parameters $\mu_{\mathbf{y}}$, $\mu_{\mathbf{z}}$, and $\sigma^2_{\mathbf{z}}$ conditioned on the target

Algorithm 1: Proposed lossy bits-back coding method (Section 4.3.3 and Figure 4.1e-f).

Global Constants: Trained hierarchical VAE with model $p(\mathbf{z}, \mathbf{y}, \mathbf{x}) = p(\mathbf{z})\, p(\mathbf{y}|\mathbf{z})\, p(\mathbf{x}|\mathbf{y})$ and inference networks $f(\,\cdot\,; \phi)$ and $f_{\mathrm{h}}(\,\cdot\,; \phi_{\mathrm{h}})$, see Figure 4.1e ($f$ is only used in subroutine `encode`).

Subroutine `encode`(image $\mathbf{x}$, side information $\boldsymbol{\xi}$) $\mapsto$ returns compressed bitstring $\mathbf{s}$

1 Initialize $\mu_{\mathbf{y}} \leftarrow f(\mathbf{x}; \phi)$ and $(\mu_{\mathbf{z}}, \sigma^2_{\mathbf{z}}) \leftarrow f_{\mathrm{h}}(\mu_{\mathbf{y}}; \phi_{\mathrm{h}})$. ▷ *Figure 4.1e (blue)*

2 Optimize over $\hat{\mathbf{y}}$ using SGA (Section 4.3.2), and over $\mu_{\mathbf{z}}$ and $\sigma^2_{\mathbf{z}}$ using BBVI. ▷ *Figure 4.1e (red)*

3 Reset $(\mu_{\mathbf{z}}, \sigma^2_{\mathbf{z}}) \leftarrow$ `reproducible_BBVI`$(\hat{\mathbf{y}})$. ▷ *See below.*

4 Decode side information $\boldsymbol{\xi}$ into $\hat{\mathbf{z}}$ using $Q(\hat{\mathbf{z}}|\mu_{\mathbf{z}}, \sigma^2_{\mathbf{z}})$ as entropy model.

5 Encode $\hat{\mathbf{z}}$ and $\hat{\mathbf{y}}$ into $\mathbf{s}$ using $P(\hat{\mathbf{z}})$ and $P(\hat{\mathbf{y}}|\hat{\mathbf{z}})$ as entropy models, respectively.

Subroutine `decode`(compressed bitstring $\mathbf{s}$) $\mapsto$ returns (lossy reconstruction $\hat{\mathbf{x}}$, side info $\boldsymbol{\xi}$)

6 Decode $\mathbf{s}$ into $\hat{\mathbf{z}}$ and $\hat{\mathbf{y}}$; then get reconstructed image $\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} p(\mathbf{x}|\hat{\mathbf{y}})$.

7 Set $(\mu_{\mathbf{z}}, \sigma^2_{\mathbf{z}}) \leftarrow$ `reproducible_BBVI`$(\hat{\mathbf{y}})$. ▷ *See below.*

8 Encode $\hat{\mathbf{z}}$ into $\boldsymbol{\xi}$ using $Q(\hat{\mathbf{z}}|\mu_{\mathbf{z}}, \sigma^2_{\mathbf{z}})$ as entropy model.

Subroutine `reproducible_BBVI`(discrete latents $\hat{\mathbf{y}}$) $\mapsto$ returns variational parameters $(\mu_{\mathbf{z}}, \sigma^2_{\mathbf{z}})$

9 Initialize $(\mu_{\mathbf{z}}, \sigma^2_{\mathbf{z}}) \leftarrow f_{\mathrm{h}}(\hat{\mathbf{y}}; \phi_{\mathrm{h}})$; seed random number generator reproducibly. ▷ *Figure 4.1f (blue)*

10 Refine $\mu_{\mathbf{z}}$ and $\sigma^2_{\mathbf{z}}$ by running BBVI for fixed $\hat{\mathbf{y}}$. ▷ *Figure 4.1f (red)*

image $\mathbf{x}$ using the trained inference networks (line 1). It then jointly performs SGA over $\hat{\mathbf{y}}$ by following Section 4.3.2, and Black-Box Variational Inference (BBVI) over $\mu_{\mathbf{z}}$ and $\sigma^2_{\mathbf{z}}$ by minimizing the the NELBO (line 2). At the end of the routine, the encoder *decodes* the provided side information $\boldsymbol{\xi}$ (an arbitrary bitstring) into $\hat{\mathbf{z}}$ using $Q(\hat{\mathbf{z}}|\mu_{\mathbf{z}}, \sigma^2_{\mathbf{z}})$ as entropy model (line 4) and then encodes $\hat{\mathbf{z}}$ and $\hat{\mathbf{y}}$ as usual (line 5).

The important step happens on line 3: up until this step, the fitted variational parameters $\mu_{\mathbf{z}}$ and $\sigma^2_{\mathbf{z}}$ depend on the target image $\mathbf{x}$ due to their initialization on line 1. This would prevent the decoder, which does not have access to $\mathbf{x}$, from reconstructing the side information $\boldsymbol{\xi}$ by encoding $\hat{\mathbf{z}}$ with the entropy model $Q(\hat{\mathbf{z}}|\mu_{\mathbf{z}}, \sigma^2_{\mathbf{z}})$. Line 3 therefore re-fits the variational parameters $\mu_{\mathbf{z}}$ and $\sigma^2_{\mathbf{z}}$ in a way that is exactly reproducible based on $\hat{\mathbf{y}}$ alone. This is done in the subroutine `reproducible_BBVI`, which performs BBVI in the prior model $p(\mathbf{z}, \mathbf{y}) =$

$p(\mathbf{z})p(\mathbf{y}|\mathbf{z})$, treating $\mathbf{y} = \hat{\mathbf{y}}$ as observed and only $\mathbf{z}$ as latent. Although we reset $\mu_{\mathbf{z}}$ and $\sigma^2_{\mathbf{z}}$ on line 3 immediately after optimizing over them on line 2, optimizing jointly over both $\hat{\mathbf{y}}$ and $(\mu_{\mathbf{z}}, \sigma^2_{\mathbf{z}})$ on line 2 allows the method to find a better $\hat{\mathbf{y}}$.

The decoder decodes $\hat{\mathbf{z}}$ and $\hat{\mathbf{y}}$ as usual (line 6). Since the subroutine `reproducible_BBVI` depends only on $\hat{\mathbf{y}}$ and uses a fixed random seed (line 9), calling it from the decoder on line 7 yields the exact same entropy model $Q(\hat{\mathbf{z}}|\mu_{\mathbf{z}}, \sigma^2_{\mathbf{z}})$ as used by the encoder, allowing the decoder to recover $\boldsymbol{\xi}$ (line 8).

4.4 Experiments

We demonstrate the empirical effectiveness of our approach by applying two variants of it —with and without bits-back coding—to an established (but not state-of-the-art) base model [Minnen et al., 2018]. We improve its performance drastically, achieving an average of over 15% BD rate savings on Kodak [1993] and 20% on Tecnick [Asuni and Giachetti, 2014], outperforming the previous state-of-the-art of both classical and neural lossy image compression methods. We conclude with ablation studies.

Proposed Methods. Table 4.1 describes all compared methods, marked as [M1]-[M9] for short. We tested two variants of our method ([M1] and [M2]): *SGA* builds on the exact same model and training procedure as in [Minnen et al., 2018] (the *Base Hyperprior* model [M3]) and changes only how the trained model is used for compression by introducing hybrid amortized-iterative inference [Campos et al., 2019] (Section 4.3.1) and Stochastic Gumbel Annealing (Section 4.3.2). *SGA+BB* adds to it bits-back coding (Section 4.3.3), which requires changing the inference model over hyperlatents $\mathbf{z}$ to admit a more flexible variational distribution $q(\mathbf{z}|\mathbf{x})$, and no longer restricts the shape of $p(\mathbf{z})$, as discussed in Section 4.3.3. The two proposed variants address different use cases: *SGA* is a "standalone" variant of our

Table 4.1: Compared methods ([M1]-[M9]) and ablations ([A1]-[A6]). We propose two variants: [M1] compresses images, and [M2] compresses images + side information via bits-back coding.

	Name	Explanation and Reference (for baselines)
ours	[M1] SGA	Proposed standalone variant: same trained model as [M3], SGA (Section 4.3.2) at compression time.
	[M2] SGA + BB	Proposed variant with bits-back coding: both proposed improvements of Sections 4.3.2, and 4.3.3.
baselines	[M3] Base Hyperprior	Base method of our two proposals, reviewed in Section 4.2 of the present paper [Minnen et al., 2018].
	[M4] Context + Hyperprior	Like [M3] but with an extra context model defining the prior $p(\mathbf{y}\|\mathbf{z})$ [Minnen et al., 2018].
	[M5] Context-Adaptive	Context-Adaptive Entropy Model proposed in [Lee et al., 2019a].
	[M6] Hyperprior Scale-Only	Like [M3] but the hyperlatents $\mathbf{z}$ model only the scale (not the mean) of $p(\mathbf{y}\|\mathbf{z})$ [Ballé et al., 2018b].
	[M7] CAE	Pioneering "Compressive Autoencoder" model proposed in [Theis et al., 2017b].
	[M8] BPG 4:4:4	State-of-the-art classical lossy image compression codec 'Better Portable Graphics' [Bellard, 2014].
	[M9] JPEG 2000	Classical lossy image compression method [Adams, 2001].
ablations	[A1] MAP	Like [M1] but with continuous optimization over $\mu_{\mathbf{y}}$ and $\mu_{\mathbf{z}}$ followed by rounding instead of SGA.
	[A2] STE	Like [M1] but with straight-through estimation (i.e., round only on backpropagation) instead of SGA.
	[A3] Uniform Noise	Like [M1] but with uniform noise injection instead of SGA (see Section 4.3.1) [Campos et al., 2019].
	[A4] Deterministic Annealing	Like [M1] but with deterministic version of Stochastic Gumbel Annealing
	[A5] BB without SGA	Like [M2] but without optimization over $\hat{\mathbf{y}}$ at compression time.
	[A6] BB without iterative inference	Like [M2] but without optimization over $\hat{\mathbf{y}}$, $\mu_{\mathbf{z}}$, and $\sigma^2_{\mathbf{z}}$ at compression time.

method that compresses individual images, while *SGA+BB* encodes images with additional side information. In all results, we used Adam [Kingma and Ba, 2014] for optimization, and annealed the temperature of SGA by an exponential decay schedule, and found good convergence without per-model hyperparameter tuning. We provide details in Appendix Section A.2.

Baselines. We compare to the *Base Hyperprior* model from [Minnen et al., 2018] without our proposed improvements ([M3] in Table 4.1), two state-of-the-art neural methods ([M4] and [M5]), two other neural methods ([M6] and [M7]), the state-of-the art classical codec BPG [M8], and JPEG 2000 [M9]. We reproduced the *Base Hyperprior* results, and took the other results from [Ballé et al.].

Results. Figure 4.4 compares the compression performance of our proposed method to existing baselines on the Kodak [1993] dataset, using the standard Peak Signal-to-Noise Ratio (PSNR) quality metric (higher is better), averaged over all images for each considered quality setting $\lambda \in [0.001, 0.08]$. The bitrates in our own results ([M1-3], [A1-6]) are based on rate estimates from entropy models, but do not differ significantly from actual file sizes produced by our codec implementation (which has negligible ($< 0.5\%$) overhead). The left

panel of Figure 4.4 plots PSNR vs. bitrate, and the right panel shows the resulting BD rate savings [Bjontegaard, 2001] computed relative to BPG as a function of PSNR for readability (higher is better). The BD plot cuts out CAE [M7] and JPEG 2000 [M9] at the bottom, as they performed much worse. Overall, both variants of the proposed method (blue and orange lines in Figure 4.4) improve substantially over the *Base Hyperprior* model (brown), and outperform the previous state-of-the-art. We report similar results on the Tecnick dataset [Asuni and Giachetti, 2014] in Appendix A. Figure 4.6 shows a qualitative comparison of a compressed image (in the order of BPG, *SGA* (proposed), and the *Base Hyperprior*), in which we see that our method notably enhances the baseline image reconstruction at comparable bitrate, while avoiding unpleasant visual artifacts of BPG.

Ablations. Figure 4.5 compares BD rate improvements of the two variants of our proposal (blue and orange) to six alternative choices (ablations [A1]-[A6] in Table 4.1), measured relative to the *Base Hyperprior* model [Minnen et al., 2018] (zero line; [M3] in Table 4.1), on which our proposals build. [A1]-[A4] replace the discretization method of *SGA* [M1] with four alternatives discussed at the end of Section 4.3.2. [A5] and [A6] are ablations of our proposed bits-back variant *SGA+BB* [M2], which remove iterative optimization over $\hat{\mathbf{y}}$, or over $\hat{\mathbf{y}}$ and $q(\mathbf{z})$, respectively, at compression time. Going from red [A3] to orange [A2] to blue [M1] traces our proposed improvements over [Campos et al., 2019] by adding SGA (Section 4.3.2)

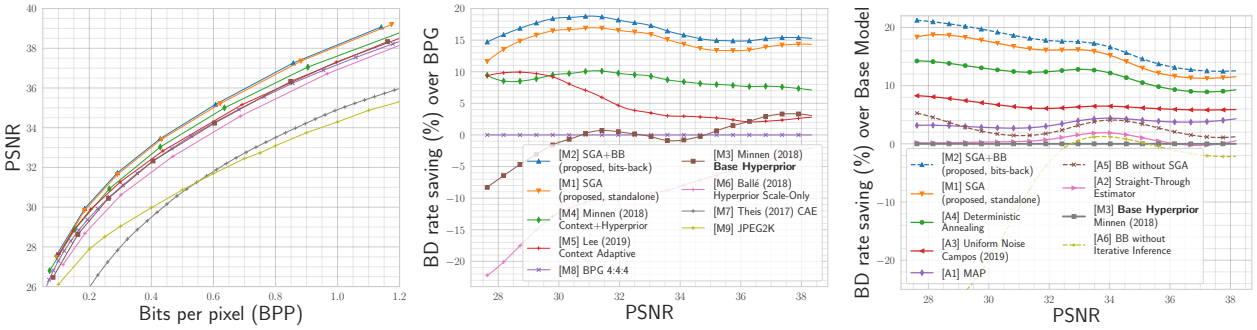


Figure 4.4: Compression performance comparisons on Kodak [1993] against existing baselines. Left: R-D curves. Right: BD rate savings (%) relative to BPG. Legend shared; higher values are better in both.

Figure 4.5: BD rate savings of various ablation methods relative to *Base Hyperprior* on Kodak.

(a) Original image (from Kodak [1993] dataset)

(b) BPG 4:4:4; 0.14 BPP (PSNR = 30.4)

(c) Proposed (SGA); 0.13 BPP (PSNR = 31.2)

(d) [Minnen et al., 2018]; 0.12 BPP (PSNR = 29.8)

Figure 4.6: Qualitative comparison of lossy compression performance. Our method (c; [M1] in Table 4.1) significantly boosts the visual quality of the *Base Hyperprior* method (d; [M3] in Table 4.1) at similar bit rates. It sharpens details of the hair and face (see insets) obscured by the baseline method (d), while avoiding the ringing artifacts around the jaw and ear pendant produced by a classical codec (b).

and then bits-back (Section 4.3.3). It shows that iterative inference (Section 4.3.1) and SGA contribute approximately equally to the performance gain, whereas the gain from bits-back coding is smaller. Interestingly, lossy bits-back coding without iterative inference and SGA actually hurts performance [A6]. As Section 4.3.3 mentioned, this is likely because bits-back coding constrains the posterior over $\mathbf{z}$ to be conditioned only on $\hat{\mathbf{y}}$ and not on the original image $\mathbf{x}$.

4.5 Discussion

Starting from the variational inference view on data compression, we proposed three enhancements to the standard inference procedure in VAEs: hybrid amortized-iterative inference, Stochastic Gumbel Annealing, and lossy bits-back coding, which translated to dramatic performance gains on lossy image compression. Improved inference provides a new promising

direction to improved compression, orthogonal to modeling choices (e.g., with auto-regressive priors [Minnen et al., 2018], which can harm decoding efficiency). Although lossy-bits-back coding in the present VAE only gave relatively minor benefits, it may reach its full potential in more hierarchical architectures as in [Kingma et al., 2019]. Similarly, carrying out iterative inference also at *training time* may lead to even more performance improvement, with techniques like iterative amortized inference [Marino et al., 2018].

Chapter 5

Asymmetrically-Powered Neural Compression for Decoding Efficiency

Most of the material in this chapter has been published in the following conference paper,

Computationally-Efficient Neural Image Compression with Shallow Decoders. Yibo Yang, Stephan Mandt. *International Conference on Computer Vision (ICCV), 2023*.

Additional results and supplementary information can be found in Appendix B, and the project repo can be found at `https://github.com/mandt-lab/shallow-ntc`.

A word on notation: this chapter largely uses the same notation as in Section 3.2.1 (nonlinear transform coding).

Neural image compression methods have seen increasingly strong performance in recent years. However, they suffer orders of magnitude higher computational complexity compared to traditional codecs, which hinders their real-world deployment. This paper takes a step

forward towards closing this gap in decoding complexity by using a shallow or even linear decoding transform resembling that of JPEG. To compensate for the resulting drop in compression performance, we exploit the often asymmetrical computation budget between encoding and decoding, by adopting more powerful encoder networks and iterative encoding. We theoretically formalize the intuition behind, and our experimental results establish a new frontier in the trade-off between rate-distortion and decoding complexity for neural image compression. Specifically, we achieve rate-distortion performance competitive with the established mean-scale hyperprior architecture of Minnen et al. (2018) at less than 50K decoding FLOPs/pixel, reducing the baseline's overall decoding complexity by 80%, or over 90% for the synthesis transform alone.

5.1 Introduction

Deep-learning-based methods for data compression [Yang et al., 2023a] have achieved increasingly strong performance on visual data compression, increasingly exceeding classical codecs in rate-distortion performance [Ballé et al., 2017, Minnen et al., 2018, Yang et al., 2020b, Cheng et al., 2020, Yang et al., 2021, Mentzer et al., 2022]. However, their enormous computational complexity compared to classical codecs, especially required for decoding, is a roadblock towards their wider adoption [Minnen, 2021, Mukherjee, 2022]. In this work, inspired by the parallel between nonlinear transform coding and traditional transform coding [Duan et al., 2022], we replace deep convolutional decoders with extremely lightweight and shallow (and even linear) decoding transforms, and establish the R-D (rate-distortion) performance of neural image compression when operating at the lower limit of decoding complexity.

More concretely, our contributions are as follows:

- We offer new insight into the image manifold parameterized by learned synthesis

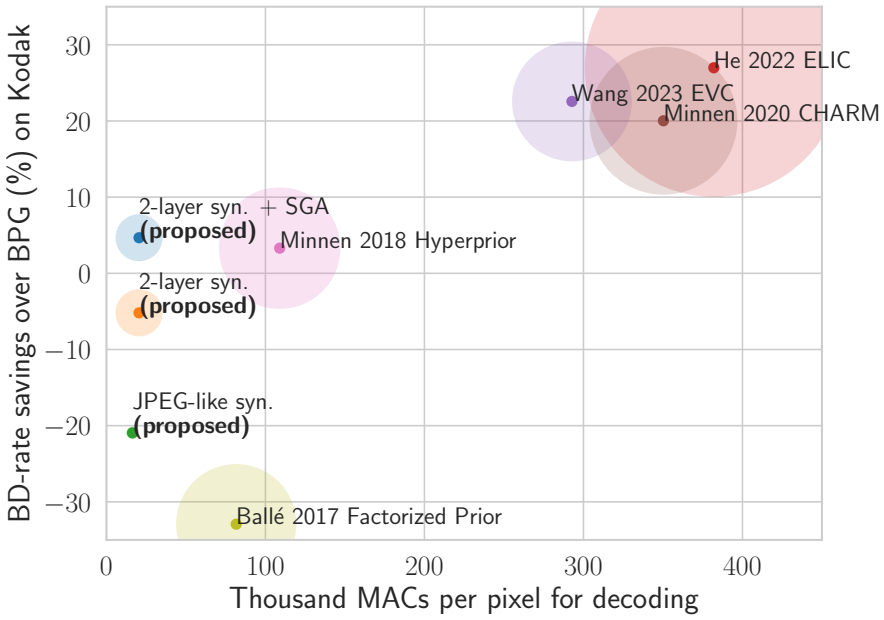


Figure 5.1: R-D performance on Kodak v.s. decoding computation complexity as measured in KMACs (thousand multiply-accumulate operations) per pixel. The circle radius corresponds to the parameter count of the synthesis transform in each method (see Table. 5.1)

transforms in nonlinear transform coding. Our results suggest that the learned manifold is relatively flat and preserves linear combinations in the latent space, in contrast to its highly nonlinear counterpart in generative modeling [Chen et al., 2020].

- Inspired by the parallel between neural image compression and traditional transform coding, we study the effect of linear synthesis transform within a hyperprior architecture. We show that, perhaps surprisingly, a JPEG-like synthesis can perform similarly to a deep linear CNN, and we shed light on the role of nonlinearity in the perceptual quality of neural image compression.

- We give a theoretical analysis of the R-D cost of neural lossy compression in an asymptotic setting, which quantifies the performance implications of varying the complexity of encoding and decoding procedures.

- We equip our JPEG-like synthesis with powerful encoding methods, and augment it with a single hidden layer. This simple approach yields a new state-of-the-art result in

the trade-off between R-D performance and decoding complexity for nonlinear transform coding, in the regime of sub-50K FLOPs per pixel believed to be dominated by classical codecs.

5.2 Background and Notation

5.2.1 Neural Image Compression

Most existing neural lossy compression approaches are based on the paradigm of nonlinear transform coding (NTC) [Ballé et al., 2021]. Traditional transform coding [Goyal et al., 2000] involves designing a pair of analysis (encoding) transform f and synthesis (decoding) transform g such that the encoded representation of the data achieves good R-D (rate-distortion) performance. NTC essentially learns this process through data-driven optimization. Let the input color image be $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$. The analysis transform computes a continuous latent representation $\mathbf{z} := f(\mathbf{x})$, which is then quantized to $\hat{\mathbf{z}} = \lfloor \mathbf{z} \rceil$ and transmitted to the receiver under an entropy model $P(\hat{\mathbf{z}})$; the final reconstruction is then computed by the synthesis transform as $\hat{\mathbf{x}} := g(\hat{\mathbf{z}})$. The hard quantization is typically replaced by uniform noise to enable end-to-end training [Ballé et al., 2016]. We refer to [Yang et al., 2023a, Section 3.3.3] as well as Section 3.2.1 of the current thesis for additional details.

Instead of orthogonal linear transforms in traditional transform coding, the analysis and synthesis transforms in NTC are typically CNNs (convolutional neural networks) [Theis et al., 2017a, Ballé et al., 2016] or variants with residual connections or attention mechanisms [Cheng et al., 2020, He et al., 2022]. The (convolutional) latent coefficients $\mathbf{z} \in \mathbb{R}^{h,w,C}$ form a 3D tensor with C channels and a spatial extent (h, w) smaller than the input image. We denote the downsampling factor by s, i.e., $s = H/h = W/w$; this is also the "upsampling" factor of the synthesis transform.

To improve the bitrate of NTC, a *hyperprior* [Ballé et al., 2018a, Minnen et al., 2018] is commonly used to parameterize the entropy model $P(\hat{\mathbf{z}})$ via another set of latent coefficients $\mathbf{h}$ and an associated pair of transforms (f_h, g_h). The hyper analysis f_h computes $\mathbf{h} = f_h(\hat{\mathbf{z}})$ at encoding time, and the hyper synthesis g_h predicts the (conditional) entropy model $P(\hat{\mathbf{z}}|\hat{\mathbf{h}})$ based on the quantized $\hat{\mathbf{h}} = \lfloor \mathbf{h} \rceil$. We adopt the Mean-scale Hyperprior from Minnen et al. [2018] as our base architecture, which is widely used as a basis for other NTC methods [Johnston et al., 2019, Cheng et al., 2020, Minnen and Singh, 2020, He et al., 2022]. In this architecture, the various transforms are parameteried by CNNs, with GDN activation [Ballé et al., 2016] being used in the analysis and synthesis transforms and ReLU activation in the hyper transforms. Importantly, the synthesis transform (g) accounts for over 80% of the overall decoding complexity (see Table 5.1), and is the focus of this work.

5.2.2 Iterative Inference

Given an image $\mathbf{x}$ to be encoded, instead of computing its discrete representation by rounding the output of the analysis transform, i.e., $\hat{\mathbf{z}} = \lfloor f(\mathbf{x}) \rceil$, Yang et al. [2020a] cast the encoding problem as that of variational inference, and propose to infer the discrete representation that optimizes the per-data R-D cost. Their proposed method, SGA (Stochastic Gumbel Annealing), essentially solves a discrete optimization problem by constructing a categorical variational distribution $q(\mathbf{z}|\mathbf{x})$ and optimizing w.r.t. its parameters by gradient descent, while annealing it to become deterministic so as to close the quantization gap [Yang et al., 2020a]. In this work, we will adopt their proposed standalone procedure and opt to run SGA at test time, essentially treating it as a powerful black-box encoding procedure for a given NTC architecture.

5.3 Methodology

We begin with new empirical insight into the qualitative similarity between the synthesis transforms in NTC and traditional transform coding [Duan et al., 2022] (Sec. 5.3.1). This motivates us to adopt simpler synthesis transforms, such as JPEG-like block-wise linear transforms, which are computationally much more efficient than deep neural networks (Sec. 5.3.2). We then analyze the resulting effect on R-D performance and mitigate the performance drop using powerful encoding methods from the neural compression toolbox (Sec. 5.3.3).

5.3.1 The Case for a Shallow Decoder

Although the transforms in NTC are generally black-box deep CNNs, Duan et al. [2022] showed that they in fact bear strong qualitative resemblance to the orthogonal transforms in traditional transform coding. They showed that the learned synthesis transform in various NTC architectures satisfy a certain separability property, i.e., a latent tensor can be decomposed spatially or across channels, then decoded separately, and finally combined in the pixel space to produce a reasonable reconstruction. Moreover, decoding "standard basis" tensors in the latent space produces image patterns resembling the basis functions of orthogonal transforms. [1]

Here, we obtain new insights into the behavior of the learned synthesis transform in NTC. We show that the manifold of image reconstructions is approximately flat, in the sense that straight paths in the latent space are mapped to approximately straight paths (i.e., naive linear interpolations) in the pixel space. Additionally, the learned synthesis transform exhibits an approximate "mixup" [Zhang et al., 2017] behavior despite the lack of such explicit

[1] We note that performing Principal Component Analysis on small image patches also results in similar patterns; see Figure B.4 in the Appendix.

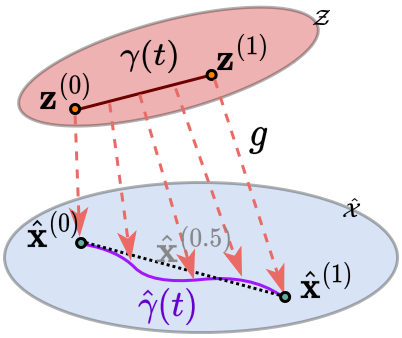


Figure 5.2: Conceptual illustration of the image manifold parameterized by $\hat{\gamma}(t)$ (purple curve), obtained by decoding a straight path $\gamma(t)$ in the latent space. We show it does not significantly deviate from a straight path (dashed line) connecting its two end points.

regularization during training.

Suppose we are given an arbitrary pair of images $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$, and we obtain their latent coefficients $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$ using the analysis transform (we ignore the effect of quantization as in [Duan et al., 2022]). Let $\gamma : [0, 1] \to \mathcal{Z}$ be the straight path in the latent space defined by the two latent tensors, i.e., $\gamma(t) := (1 - t)\mathbf{z}^{(0)} + t\mathbf{z}^{(1)}$. Using the synthesis transform g, we can then map the curve in the latent space to one in the space of reconstructed images, defined by $\hat{\gamma}(t) := g(\gamma(t))$. We denote the two end-points of the curve by $\hat{\mathbf{x}}^{(0)} := g(\mathbf{z}^{(0)}) = \hat{\gamma}(0)$ and $\hat{\mathbf{x}}^{(1)} := g(\mathbf{z}^{(1)}) = \hat{\gamma}(1)$. Instead of traversing the image manifold parameterized by g, we could also travel between the two end-points in a straight path, which we define by $\hat{\mathbf{x}}^{(t)} := (1 - t)\hat{\mathbf{x}}^{(0)} + t\hat{\mathbf{x}}^{(1)}$ and is given by a simple linear interpolation in the pixel space. The setup is illustrated in Figure 5.2.

Fig. 5.3 visualizes an example of the resulting curve of images $\hat{\gamma}(t)$ (top row), compared to the interpolating straight path $\hat{\mathbf{x}}^{(t)}$ (bottom row), as t goes from 0 to 1. The results appear very similar, suggesting the latent coefficients largely carry local and mostly low-level information about the image signal. As a rough measure of the deviation between the two trajectories, Fig. 5.4a computes the MSE between $\hat{\gamma}(t)$ and $\hat{\mathbf{x}}^{(t)}$ at corresponding time steps, for pairs of random image crops from COCO [Lin et al., 2014]. The results (solid lines) indicate that

Figure 5.3: Visualizing the 1-D manifold of image reconstructions $\{\hat{\gamma}(t)|t \in [0,1]\}$ (**top row**) and the linear interpolation between its two end points, $\{(1-t)\hat{\mathbf{x}}^{(0)} + t\hat{\mathbf{x}}^{(1)}|t \in [0,1]\}$ (**bottom row**).

the two curves do not align perfectly. However, since the parameterization of any curve is not unique, we get a better sense of the behavior of the manifold curve $\hat{\gamma}(t)$ by considering its *length* $L(\hat{\gamma})$ in relation to the *length* of the interpolating straight path $\|\hat{\mathbf{x}}^{(0)} - \hat{\mathbf{x}}^{(1)}\|$. We compute the two lengths (the curve length can be computed using the Jacobian of g; see Appendix Sec. B.4), and plot them for random image pairs in Fig. 5.4b. The resulting curve lengths fall very closely to the straight path lengths regardless of the absolute length of the curves, indicating that the curves globally follow nearly straight paths. Note that if g was linear (affine), then $\hat{\gamma}(t)$ and $\hat{\mathbf{x}}^{(t)}$ would perfectly overlap.

Additionally, inspired by *mixup* regularization [Zhang et al., 2017], we examine how well the synthesized curve $\hat{\gamma}(t)$ can reconstruct the linear interpolation of the two *ground truth* images, defined by $\mathbf{x}^{(t)} := (1-t)\mathbf{x}^{(0)} + t\mathbf{x}^{(1)}$. Fig. 5.4a plots the reconstruction error for the same random image pairs in dashed lines, and shows that the synthesized curve $\hat{\gamma}(t)$ generally offers consistent reconstruction quality along the entire trajectory. Note that if g was linear (affine), then this reconstruction error would vary linearly across t.

The above observations form a stark contrast to the typical behavior of the decoder network in generative modeling, where different images tend to be separated by regions of low density

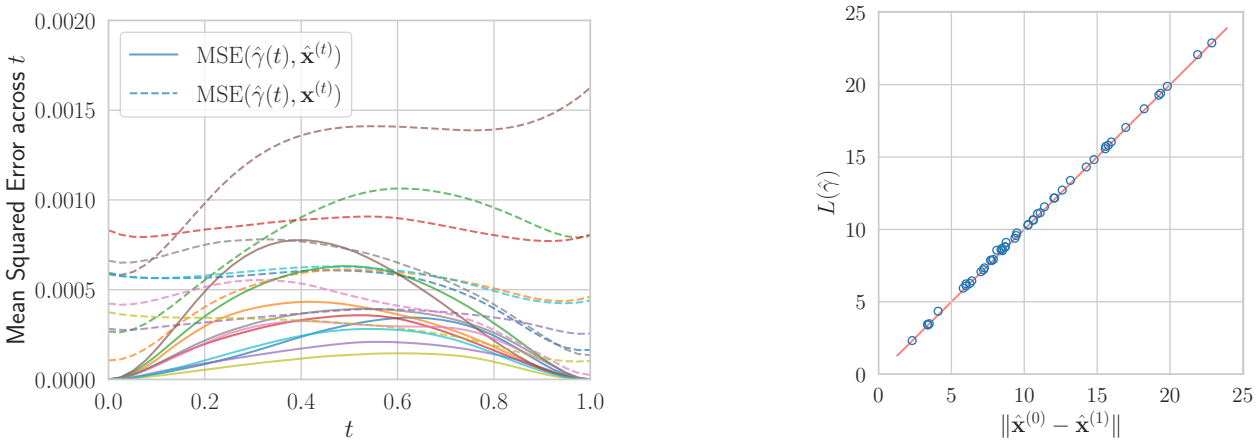


Figure 5.4: The effect of traversing the synthesis manifold, with end points defined by random image pairs. **(a)**: Mean-squared error distance between the decoded curve $\hat{\gamma}(t)$ and straight paths in the image space (reconstructions $\hat{\mathbf{x}}^{(t)}$ and originals $\mathbf{x}^{(t)}$). **(b)**: The length of the curve $\hat{\gamma}$ v.s. that of the interpolating straight path $\hat{\mathbf{x}}^{(t)}$. The image pixel values are scaled to $[-0.5, 0.5]$.

under the model, and the decoder function varies rapidly when crossing such boundaries [Chen et al., 2018b], e.g., across a linear interpolation of images in pixel space.

We obtained these results with a Mean-scale Hyperprior model [Minnen et al., 2018] trained with $\lambda = 0.01$, and we observe similar behavior at other bit-rates (with the curves $\hat{\gamma}$ becoming even "straighter" at higher bit-rates) and in various NTC architectures [Ballé et al., 2017, Minnen et al., 2018, Minnen and Singh, 2020] (see Appendix Sec. B.4 for more examples). Our empirical observations corroborate the earlier findings [Duan et al., 2022], and raise the question: Given the many similarities, can we replace the deep convolutional synthesis in NTC with a linear (affine) function? Our motivation is mainly computational: a linear synthesis can offer drastic computation savings over deep neural networks. This is not necessarily the case for an arbitrary linear (affine) function from the latent to image space, so we restrict ourselves to efficient convolutional architectures. As we show empirically in Sec. 5.4.3, a single JPEG-like transform with a large enough kernel size can emulate a more general cascade of transposed convolutions, while being much more computationally efficient. Compared to fixed and orthogonal transforms in traditional transform coding, learning a linear synthesis

from data allows us to still benefit from end-to-end optimization. Further, in Sec. 5.4.2, we show that strategically incorporating a small amount of nonlinearity can significantly improve the R-D performance without much increase in computation complexity.

5.3.2 Shallow Decoder Design

JPEG-like synthesis At its core, JPEG works by dividing an input image into 8×8 blocks and applying block-wise linear transform coding. This can be implemented efficiently in hardware and is a key factor in JPEG's enduring popularity. By analogy to JPEG, we interpret the $h \times w \times C$ latent tensor in NTC as the coefficients of a linear synthesis transform. In the most basic form, the output reconstructions are computed in $s \times s$ blocks, similarly to JPEG. Specifically, the (i, j)th block reconstruction is computed as a linear combination of (learned) "basis images" $\mathbf{K}_c \in \mathbb{R}^{s \times s \times C_{out}}, c = 1, ..., C$, weighted by the vector of (quantized) coefficients $\mathbf{z}_{i,j} \in \mathbb{R}^C$ associated with the (i, j)th spatial location:

$$\hat{B}_{i,j} = \sum_{c=1}^{C} \mathbf{z}_{i,j,c} \mathbf{K}_c. \tag{5.1}$$

Note that we recover the per-channel discrete cosine transform of JPEG by setting $s = 8, C = 64, C_{out} = 1$, and $\{\mathbf{K}_c, c = 1, ..., 64\}$ to be the bases of the 8×8 discrete cosine transform. Eq 5.1 can be implemented efficiently via a transposed convolution on $\mathbf{z}$, using $\mathbf{K}$ as the kernel weights and s as the stride. In terms of MACs, the computation complexity of the JPEG-like synthesis then equals[2]

$$M(\text{JPEG-like}) = C \times h \times w \times s^2 \times C_{out}, \tag{5.2}$$

where $C_{out} = 3$ for a color image. Note that for a given latent tensor and "upsampling" rate

[2]When the latent coefficients are sparse (which often occurs at low bit-rates), this computation complexity can be further reduced by using sparse matrix/tensor operations. We leave this to future work.

s, Eq. 5.2 gives the *minimum achievable* MACs by any non-degenerate synthesis transform based on (transposed) convolutions. As we see in Sec. 5.4.2, although the minimal JPEG-like synthesis drastically reduces the decoding complexity, it can introduce severe blocking artifacts since the blocks are reconstructed independently. We therefore allow overlapping basis functions with spatial extent $k \times k$, where $k \geq s$ and $k - s$ is the number of overlapping pixels; we compute each $k \times k$ blocks as in Eq. 5.1, then form the reconstructed image by taking the sum of the (overlapping) blocks. This corresponds to simply increasing the kernel size from (s, s) to (k, k) in the corresponding transposed convolution, and increases the s^2 factor in Eq. 5.2 to k^2.

Two-layer nonlinear synthesis Despite its computational efficiency, the JPEG-like synthesis can be overly restrictive. Indeed, nonlinear transform coding benefits from the ability of the synthesis transform to adapt to the shape of the data manifold [Ballé et al., 2021]. We therefore introduce a small degree of nonlinearity in the JPEG-like transform. Many possibilities exist, and we found that introducing a single hidden layer with nonlinearity to work well. Concretely, we use two layers of transposed convolutions ($\texttt{conv\_1}, \texttt{conv\_2}$), with strides (s_1, s_2), kernel sizes (k_1, k_2), and output channels (N, C_{out}) respectively. At lower bit-rates, we found it more parameter- and compute-efficient to also allow a residual connection from $\mathbf{z}$ to the hidden activation using another transposed convolution `conv_res` (see a diagram and more details in Appendix Sec. B.1). Thus, given a latent tensor $\mathbf{z} \in \mathbb{R}^{h,w,C}$ the output is $g(\mathbf{z}) = \texttt{conv\_2}(\xi(\texttt{conv\_1}(\mathbf{z})) + \texttt{conv\_res}(\mathbf{z}))$, where ξ is a nonlinear activation.

The MAC count in this architecture is then approximately

$$\begin{aligned} M(\text{2-layer}) &= C \times h \times w \times k_1^2 \times 2N \\ &\quad + N \times hs_1 \times ws_1 \times k_2^2 \times C_{out}. \end{aligned} \tag{5.3}$$

To keep this decoding complexity low, we use large convolution kernels ($k_1 = 13$) with

aggressive upsampling ($s_1 = 8$) in the first layer, in the spirit of a JPEG-like synthesis, followed by a lightweight output layer with a smaller upsampling factor ($s_2 = 2$) and kernel size ($k_2 = 5$). We use the simplified (inverse) GDN activation [Johnston et al., 2019] for ξ as it gave the best R-D performance with minor computational overhead. We discuss these and other architectural choices in Sec. 5.4.4.

5.3.3 Formalizing the Role of the Encoder in Lossy Compression Performance

Here, we analyze the rate-distortion performance of neural lossy compression in an idealized, asymptotic setting. Our novel decomposition of the R-D objective pinpoints the performance loss caused by restricting to a simpler (e.g., linear) decoding transform, and suggests reducing the inference gap as a simple and theoretically principled remedy.

Consider a general neural lossy compressor operating as follows. Let $\mathcal{Z}$ be a latent space, $p(\mathbf{z})$ a prior distribution over $\mathcal{Z}$ known to both the sender and receiver, and $g : \mathcal{Z} \to \hat{\mathcal{X}}$ the synthesis transform belonging to a family of functions $\mathcal{G}$. Given a data point $\mathbf{x}$, the sender computes an inference distribution $q(\mathbf{z}|\mathbf{x})$; this can be the output of an encoder network, or a more sophisticated procedure such as iterative optimization with SGA [Yang et al., 2020a]. We assume relative entropy coding [Cuff, 2008b, Theis and Yosri, 2021] is applied with minimal overhead, so that the sender can send a sample of $\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})$ with an average bit-rate not much higher than $KL(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$ [Flamich et al., 2020b]. Given a neural compression method, which can be identified with the tuple $(q(\mathbf{z}|\mathbf{x}), g, p(\mathbf{z}))$, its R-D cost on data distributed according to $P_{\mathbf{X}}$ thus has the form of a negative ELBO [Flamich et al.,

2020b]

$$\mathcal{L}(q(\mathbf{z}|\mathbf{x}), g, p(\mathbf{z})) := \tag{5.4}$$
$$\lambda \mathbb{E}_{\mathbf{x}\sim P_{\mathbf{X}}, \mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}[\rho(\mathbf{x}, g(\mathbf{z}))] + \mathbb{E}_{\mathbf{x}\sim P_{\mathbf{X}}}[KL(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))],$$

where $\lambda \geq 0$ controls the R-D tradeoff, and $\rho : \mathcal{X} \times \hat{\mathcal{X}} \to [0, \infty)$ is the distortion function (commonly the MSE). Note that the encoding distribution $q(\mathbf{z}|\mathbf{x})$ appears in both the rate and distortion terms above. We show that the compression cost admits the following alternative decomposition, where the effects of $p(\mathbf{z})$, g, and $q(\mathbf{z}|\mathbf{x})$ can be isolated:

$$\mathcal{L}(q(\mathbf{z}|\mathbf{x}), g, p(\mathbf{z})) := \tag{5.5}$$
$$= \underbrace{\mathcal{F}(\mathcal{G})}_{\text{irreducible}} + \underbrace{\left(\mathbb{E}_{\mathbf{x}\sim P_{\mathbf{X}}}[-\log \Gamma_{g,p(\mathbf{z})}(\mathbf{x})] - \mathcal{F}(\mathcal{G})\right)}_{\text{modeling gap}}$$
$$+ \underbrace{\mathbb{E}_{\mathbf{x}\sim P_{\mathbf{X}}}[KL(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}|\mathbf{x})]}_{\text{inference gap}}. \tag{5.6}$$

The full derivation and definitions of various quantities are given in Sec. B.2, and mirror a similar decomposition in lossless compression [Zhang et al., 2022]; here we give a high-level explanation of the three terms. The first term represents the fundamentally irreducible cost of compression; this depends only on the intrinsic compressibility of the data $P_{\mathbf{X}}$ [Yang and Mandt, 2022] and the transform family $\mathcal{G}$. The second term represents the excess cost of compression given our particular choice of decoding architecture, i.e., the prior $p(\mathbf{z})$ and transform g, compared to the optimum achievable (the first term); we thus call it the *modeling gap*. Note that for each choice of $(g, p(\mathbf{z}))$, the optimal inference distribution has an explicit formula, which allows us to write the R-D cost under optimal inference in the form of a negative log partition function (the $-\log \Gamma$ term). Finally, we consider the effect of suboptimal inference and isolate it in the third term, representing the overhead caused by a sub-optimal encoding/inference method $q(\mathbf{z}|\mathbf{x})$ for a given model $(g, p(\mathbf{z}))$; we call it the *inference gap*.

Although the above result is derived in an asymptotic setting, it still gives us insight about the performance of neural lossy compression at varying decoder complexity. When we use a simpler synthesis transform architecture, we place restrictions on our transform family $\mathcal{G}$, thus causing the first (irreducible) part of compression cost to increase. The modeling gap may or may not increase as a result,[3] but we can always lower the overall compression cost by reducing the inference gap, without affecting the decoding computational complexity.

In this work, we explore two orthogonal approaches for reducing the inference gap, which can be further decomposed into an (1) *approximation gap* and (2) *amortization gap* [Cremer et al., 2018]. Correspondingly, for a given decoding architecture, we propose to reduce (1) by using a more powerful analysis transform, e.g., from a recent SOTA method such as ELIC [He et al., 2022], and reduce (2) by performing iterative encoding using SGA [Yang et al., 2020a] at compression time.

5.4 Experiments

5.4.1 Data and Training

We train all of our models on random 256×256 image crops from the COCO 2017 [Lin et al., 2014] dataset. We follow the standard training procedures as in [Ballé et al., 2017, Minnen et al., 2018] and optimize for MSE as the distortion metric. We verified that our base Mean-Scale Hyperprior model matches the reported performance in the original paper [Minnen et al., 2018].

[3]The modeling gap can be reduced by adopting a more expressive prior $p(\mathbf{z})$, although doing so can lead to higher decoding complexity.

Method	Computational complexity (KMAC)						Syn. param count (Mil.)	BD rate savings (%) ↑
	f	f_h	enc. tot.	g	g_h	dec. tot.		
[He et al., 2022] ELIC	255.42	6.73	262.15	255.42	126.57	381.99	7.34	26.98
[Minnen and Singh, 2020] CHARM	93.79	5.90	99.70	93.79	256.51	350.30	4.18	20.02
[Wang et al., 2023] EVC	263.25	1.86	265.11	257.94	34.82	292.76	3.38	22.56
[Minnen et al., 2018] Hyperprior	93.79	6.73	100.52	93.79	15.18	108.97	3.43	3.30
[Ballé et al., 2017] Factorized Prior	81.63	0.00	81.63	81.63	0.00	81.63	3.39	-32.93
2-layer syn. + SGA (proposed)	255.42	6.73	$\sim 10^5$	**5.34**	15.18	20.52	**1.30**	4.67
2-layer syn. (proposed)	255.42	6.73	262.15	**5.34**	15.18	20.52	**1.30**	-5.19
JPEG-like syn. (proposed)	255.42	6.73	262.15	**1.22**	15.18	16.39	**0.31**	-20.95

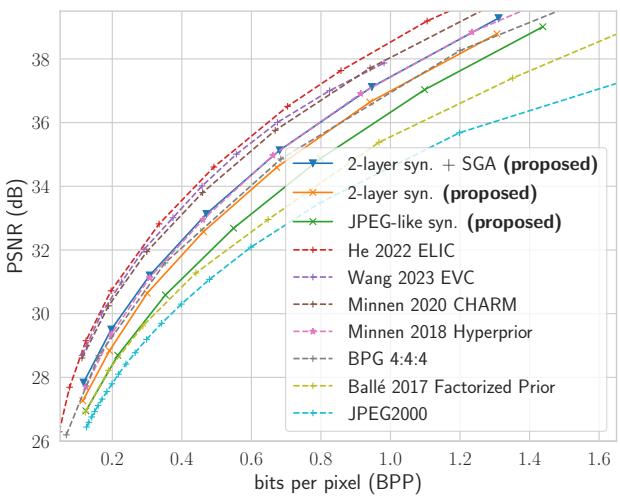
Table 5.1: Computational complexity of various neural compression methods, v.s. average BD rate savings relative to BPG [Bellard, 2014] on Kodak. Complexity is measured in KMACs (thousand multiply–accumulate operations) per pixel, and does not include entropy coding. f, f_h, g, g_h stand for analysis, hyper analysis, synthesis, and hyper synthesis transforms. We also report the parameter count of synthesis transforms (g) in the second-to-last column, and a rough estimate of the overall encoding complexity of SGA-based encoding ($\sim 10^5$ KMACs/pixel).

5.4.2 Comparison with Existing Methods

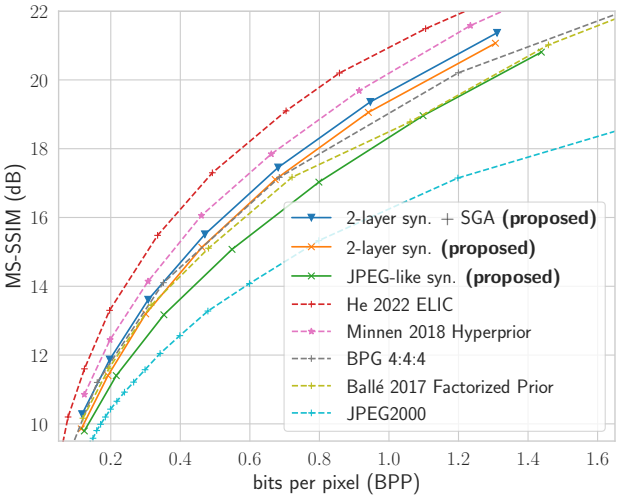
We compare our proposed methods with standard neural compression methods [Ballé et al., 2017, Minnen et al., 2018, Minnen and Singh, 2020] and state-of-the-art methods [He et al., 2022, Wang et al., 2023] targeting computational efficiency. We obtain the baseline results from the CompressAI library [Bégaint et al., 2020], or trace the results from papers when they are not available. For our shallow synthesis transforms, we use $k = 18$ in the JPEG-like synthesis, and $N = 12, k_1 = 13, k_2 = 5$ in the 2-layer synthesis; we ablate on these choices in Sections 5.4.3 and 5.4.4.

Table 5.1 summarizes the computational complexity of various methods, ordered by decreasing overall decoding complexity. We use the `keras-flops` package [4] to measure the FLOPs on 512×768 images, and report the results in KMACs (thousand multiply-accumulates) per pixel. Note that the Factorized Prior architecture [Ballé et al., 2017] lacks the hyperprior, while CHARM [Minnen and Singh, 2020] and ELIC [He et al., 2022] use autoregressive computation in the hyperior. Our proposed models borrow the same hyperprior from Mean-Scale Hyperprior [Minnen et al., 2018].

[4] `https://pypi.org/project/keras-flops/`



(a) R-D performance on Kodak; quality measured in PSNR (the higher the better).



(b) R-D performance on Kodak; quality measured in MS-SSIM dB (the higher the better).

Figure 5.5: Comparison of the R-D performance of the proposed methods with existing neural image compression methods. All the models were optimized for MSE distortion.

While most existing methods use analysis and synthesis transforms with symmetric computational complexity, our proposed methods adopt the relatively more expensive analysis transform from ELIC [He et al., 2022] (column "f"), and drastically reduces the complexity of the synthesis transform (column "g") — over 50 times smaller than in ELIC, and 17 smaller than in Mean-Scale Hyperprior.[5] As a result, the hyper synthesis transform (the same as in Mean-Scale Hyperprior) accounts for a great majority of our overall decoding complexity.

In Fig. 5.5a, we plot the R-D performance of various methods on the Kodak [1993] benchmark, with quality measured in PSNR. We also compute the BD [Bjontegaard, 2001] rate savings (%) relative to BPG [Bellard, 2014], and summarize the average BD rate savings v.s. the total decoding complexity in Table 5.1 and Fig. 5.1. As can be seen, our model with ELIC analysis transform and JPEG-like synthesis transform (**green**) comfortably outperforms the Factorized Prior architecture [Ballé et al., 2017]; the latter employs a more expensive CNN synthesis transform but a less powerful entropy model. However, our JPEG-like synthesis still significantly lags behind BPG and the Mean-Scale Hyperprior. By adopting the two-

[5] In our preliminary measurements, this translates to 6 $\sim$ 12 times reduction in running time of the synthesis transform compared to the Mean-Scale Hyperprior, depending the hardware used.

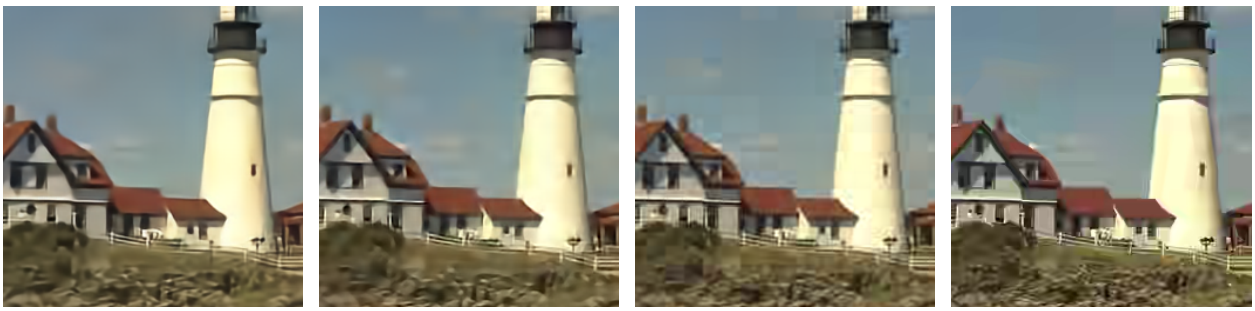


Figure 5.6: Visualizing the different kinds of distortion artifacts at comparable low bit-rates between various methods. Left to right: Mean-Scale Hyperprior [Minnen et al., 2018], two-layer synthesis (proposed), JPEG-like synthesis (proposed), and BPG [Bellard, 2014]. See Sec. 5.4.2 for relevant discussion.

layer synthesis (**orange**), the overall decoding complexity increases marginally (since the majority of complexity comes from the hyper decoder), while the R-D performance improves significantly, to within $\leq 6\%$ bit-rate of BPG. Finally, performing iterative encoding with SGA (**blue**) gives a further boost in R-D performance, outperforming the Mean-Scale Hyperprior (and BPG) without incurring any additional decoding complexity.

Additionally, we examine the R-D performance using the more perceptually relevant MS-SSIM metric [Wang et al., 2003]. Following standard practice, we display it in dB as $-10\log_{10}(1 - \text{MS-SSIM})$. The results are shown in Fig. 5.5b. We observe largely the same phenomenon as before under PSNR, except that the existing methods based on CNN decoders achieve relatively much stronger performance compared to traditional codecs such as BPG and JPEG 2000. Our proposed method with a two-layer synthesis and iterative encoding (**blue**) still outperforms BPG, but no longer outperforms the Mean-Scale Hyperprior (**pink**). Indeed, as we see in Fig. 5.6, the reconstructions of the proposed shallow synthesis transforms can exhibit artifacts similar to classical codecs (e.g., BPG) at low bit-rates, such as blocking or ringing, but to a lesser degree with the nonlinear two-layer synthesis (second panel) than the JPEG-like synthesis (third panel).

In Sec. B.4 of the Appendix, we report additional R-D results evaluated on Tecnick [Asuni and Giachetti, 2014] and the CLIC validation set [CLIC, 2018], as well as under the perceptual

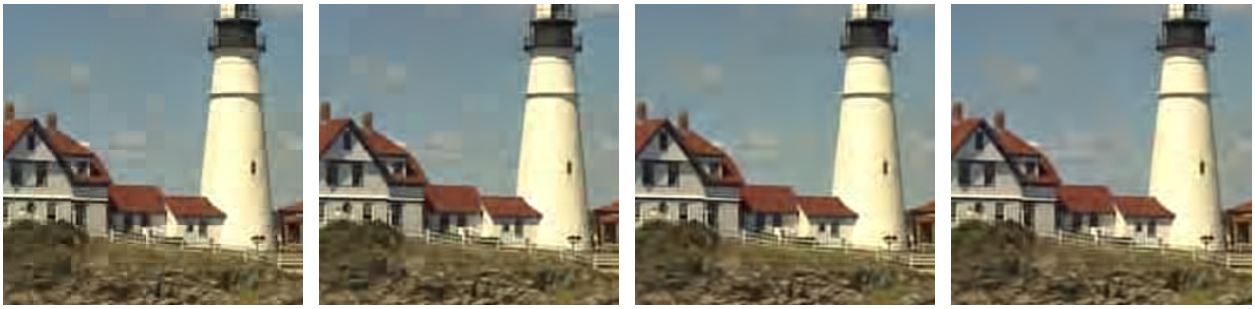


Figure 5.7: Comparing the distortion artifacts at low bit-rate for different kernel sizes ($k = 16, 18, 32$, from left to right) in our JPEG-like synthesis, as well as a linear CNN synthesis (rightmost panel). The JPEG-like blocking artifacts are reduced as k increases; see Sec. 5.4.3.

distortion LPIPS [Zhang et al., 2018]. Overall, we find that our proposed two-layer synthesis with SGA encoding matches the Hyperprior performance when evaluated on PSNR, but under-performs by $8\% \sim 12\%$ (in BD-rate) when evaluated on either MS-SSIM or LPIPS.

5.4.3 JPEG-Like Synthesis

In this section, we study the JPEG-like synthesis in isolation. We start with the Mean-Scale Hyperprior architecture, and replace its CNN synthesis with a single transposed convolution with varying kernel sizes. Additionally, instead of replacing the CNN synthesis entirely, we also consider a linear version of it ("linear CNN synthesis") where we remove all the nonlinear activation functions. This results in a composition of four transposed convolution layers, which in general cannot be expressed by a single transposed convolution; however, note that this is still a linear (afffine) map from the latent space to image space.

Fig. 5.7 illustrates the distortion artifacts of the JPEG-like synthesis and linear CNN synthesis at comparable bit-rates, and reveals the following: (i). Using the smallest non-degenerate kernel size ($k = s = 16$) results in severe blocking artifacts, e.g., as seen in the 16×16 cloud patches in the sky, similarly to JPEG. (ii). Increasing k by a small amount ($16 \rightarrow$

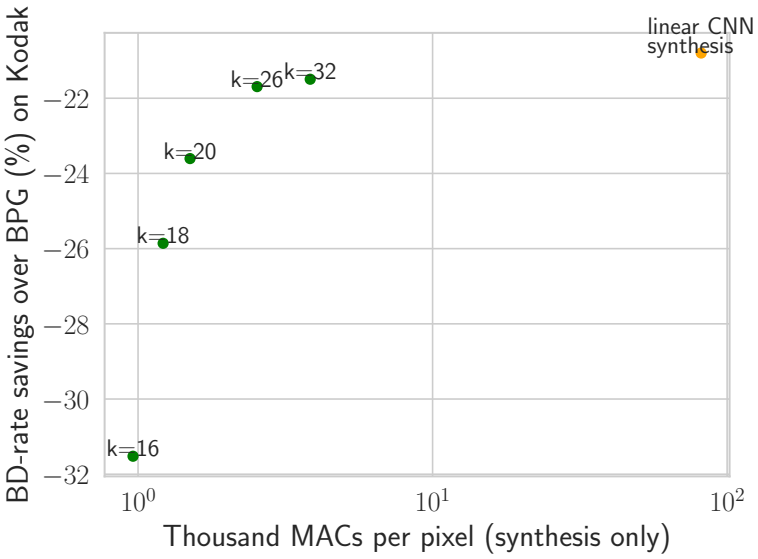


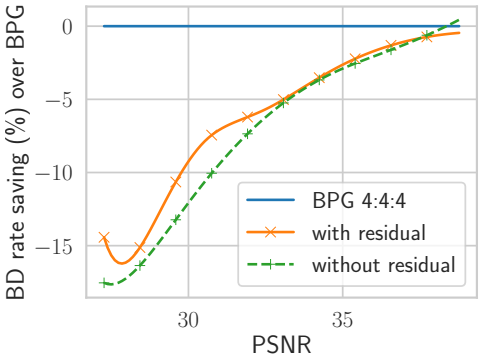
Figure 5.8: Effect of increasing kernel size (k) on the performance of JPEG-like synthesis. See Sec. 5.4.3 for details.

18) already helps smooth out the blocking, but further increase gives diminishing returns. (iii). At $k = 32$, the reconstruction of JPEG-like synthesis no longer shows obvious blocking artifacts, but shows ringing artifacts near object boundaries instead; the reconstruction by the linear CNN synthesis gives visually very similar results.

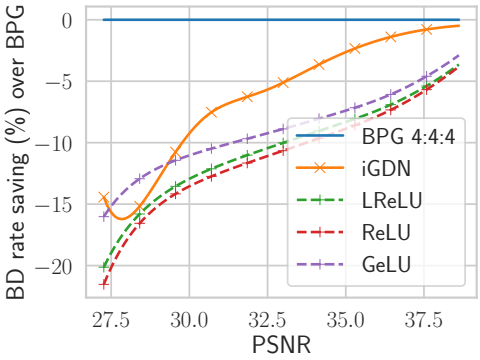
Indeed, Fig. 5.8 confirms that increasing k quantitatively improves the R-D performance of the JPEG-like synthesis, with $k = 26$ approaching the R-D performance of the linear CNN synthesis (within 1% aggregate bit-rate) while requiring 94% less FLOPs. We conclude that for image compression, a single transposed convolution with large enough kernel size can largely emulate a deep but linear CNN in PSNR performance, and the additional nonlinearity is necessary for the superior perceptual quality of nonlinear transform coding.

5.4.4 Ablation Studies

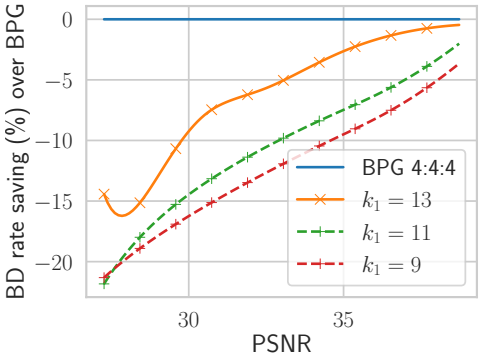
The analysis transform. We ablate on the choice of analysis transform for our proposed two-layer synthesis architecture. Replacing the analysis transform of ELIC [He et al., 2022]



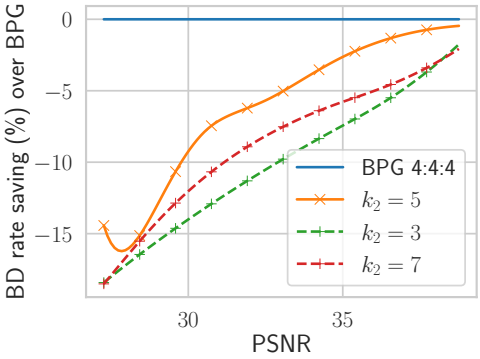
(a) Residual connection.



(b) Nonlinear activation.



(c) Hidden layer kernel size.



(d) Output layer kernel size.

Figure 5.9: Ablation on various architectural choices of the proposed two-layer synthesis transform. BD-rate savings are evaluated on Kodak (the higher the better). See Sec. 5.4.4 for a discussion.

with that of Mean-Scale Hyperprior results in over 6% worse bitrate (with BPG as the anchor). This gap can be reduced to $\sim 5\%$ by increasing the number of base channels in the CNN analysis, although with diminishing returns and becomes suboptimal compared to switching to the ELIC analysis transform. See Appendix Sec. B.4 for details.

Two-layer synthesis architecture Due to resource constraints, we were not able to conduct an exhaustive architecture search, and instead set the hyperparameters manually.

Fig. 5.9 presents ablation results on the main architectural elements of the proposed two-layer synthesis. We found that the residual connection slightly improves the R-D performance at low bit-rates, compared to a simple two-layer architecture with comparable FLOPs (using

$2N = 24$ hidden channels). We also found the use of (inverse) GDN activation [Ballé et al., 2016] and increased kernel size in the output layer (k_2) to be beneficial, which only cost a minor (less than 5%) increase in FLOPs . The number of channels (N) and kernel size (k_1) in the hidden layer are more critical in the trade-off between decoding FLOPs and R-D performance, and we leave a more detailed architecture search to future work.

5.5 Related Work

Computationally efficient neural compression To reduce the high decoding complexity in neural compression, Johnston et al. [2019] proposed to prune out filters in convolutional decoders with group-Lasso regularization [Gordon et al., 2018]. Rippel and Bourdev [2017] developed one of the earliest lossy neural image codecs with comparable running time to classical codecs, based on a multi-scale autoencoder architecture inspired by wavelet transforms such as JPEG 2000. Recent works propose computationally efficient neural compression architectures based on residual blocks [Cheng et al., 2020, He et al., 2022], more lightweight entropy model [He et al., 2021, 2022], network distillation [Wang et al., 2023], and accelerating learned entropy coding [Liu et al., 2022]. We note there is also related effort on improving the compression performance of traditional codecs with learned components while maintaining high computational efficiency [Duong et al., 2023, Isik et al., 2023].

Test-time / encoding optimization in compression The idea of improving compression performance with a powerful, content-adaptive encoding procedure is well-established in data compression. Indeed, vector quantization [Gersho and Gray, 2012] can be seen as implementing the most basic and general form of an optimization-based encoding procedure, and can be shown to be asymptotically optimal in rate-distortion performance [Cover, 1999]. The encoders in commonly used traditional codecs such as H.264 [Sullivan et al., 2004] and

HEVC [Sze et al., 2014] are also equipped with an exhaustive search procedure to select the optimal block partitioning and coding modes for each image frame. More recently, the idea of iterative and optimization-based encoding is becoming increasingly prominent in nonlinear transform coding [Campos et al., 2019, Yang et al., 2020a, van Rozendaal et al., 2021], as well as computer vision in the form of implicit neural representations [Park et al., 2019, Mildenhall et al., 2021]. It is therefore interesting to see whether ideas from vector quantization and implicit neural representations may prove fruitful for further reducing the decoding complexity in NTC.

Manifold/metric learning A distantly related line of work is in metric learning with deep generative models, where the idea is to learn a latent representation of the data such that distance in the latent space preserves the similarity in the data space. Chen et al. [Chen et al., 2018b] proposes the use of the Riemannian distance metric induced by a decoding transform of a latent variable to measure similarity in the data space. Further, they proposed to learn flat manifolds with VAEs [Chen et al., 2020], whose decoder essentially captures the geodesic distance between data points in terms of the Euclidean distance between their representations in the latent space. Their method is based on regularizing the Jacobian $\mathbf{J}$ of the decoder such that $\mathbf{J}^T\mathbf{J} \propto \mathbf{I}$, resulting in a length-preserving decoder and a latent space with low curvature, similar to what we observe with learned synthesis transforms in Section 5.3.1.

5.6 Discussion

In this work, we took a step towards closing the enormous gap between the decoding complexity of neural and traditional image compression methods. The main idea is to exploit the often asymmetrical computation budget of encoding and decoding: by pairing

a lightweight decoder with a powerful encoder, we can obtain high R-D performance while enjoying low decoding complexity. We formalize this intuition theoretically, and show that the encoding procedure affects the R-D cost of lossy compression via an inference gap, and more powerful encoders improve R-D performance by reducing this gap. In our implementation, we adopt shallow decoding transforms inspired by classical codecs such as JPEG and JPEG 2000, while employing more sophisticated encoding methods including iterative inference. Empirically, we show that by pairing a powerful encoder with a shallow decoding transform, the resulting method achieves R-D performance competitive with BPG and the base Mean-Scale Hyperprior architecture [Minnen et al., 2018], while reducing the complexity of the synthesis transform by over an order of magnitude. We suspect that the synthesis complexity can be further reduced by going beyond the transposed convolutions used in this work, e.g., via sub-pixel convolution [Shi et al., 2016] or (transposed) depthwise convolution, as well as by exploiting the sparsity [Mallat, 2008] of the transform coefficients especially at low bit-rates.

The success of nonlinear transform coding [Ballé et al., 2021] over traditional transform coding can be mostly attributed to (1) data-adaptive transforms and (2) expressive deep entropy models. We focused on improving the R-D-Compute efficiency of the synthesis transform, given that it accounts for the vast majority of decoding complexity in existing approaches, and left the hyperprior [Minnen et al., 2018] unchanged. As a result, entropy decoding (via the hyper-synthesis transform) now takes up a majority (50 % - 80%) of the overall decoding computation in our method. Interestingly, we note that related work on flat manifolds found it necessary to use an expressive prior to learn a distance-preserving decoding transform [Chen et al., 2020], and recent work in video compression [Mentzer et al., 2022] also features a simplified transform in the data space and a more expressive and computationally expensive entropy model. Given recent advances in computationally efficient entropy models [He et al., 2022, 2021, Liu et al., 2022], we are optimistic that the entropy decoder in our approach can be significantly improved in rate-distortion-complexity, and leave this important direction to

future work.

A limitation of our shallow synthesis is its worse performance on perceptual distortion compared to deeper architectures. Our study focused on the MSE distortion as in traditional transform coding; in this setting, it is known that an orthogonal linear transform gives optimal R-D performance for Gaussian-distributed data [Gersho and Gray, 2012]. However, the distribution of natural images is far from Gaussian, and compression methods are increasingly evaluated on perceptual metrics such as MS-SSIM [Wang et al., 2003] — both factors motivating the use of nonlinear transforms. We believe insights from signal processing and deep generative modeling may inspire more efficient nonlinear transforms with high perceptual quality, or an efficient pipeline based on a cheap MSE-optimized reconstruction followed by generative artifact removal/denoising for good perceptual quality.

Chapter 6

Neural Network-Based Sandwich Bounds on the Rate-Distortion Function

Most of the material in this chapter has been published in the following conference paper,

Towards Empirical Sandwich Bounds on the Rate-Distortion Function. Yibo Yang, and Stephan Mandt. *International Conference on Learning Representations (ICLR), 2022.*

Additional results and supplementary information can be found in Appendix C, and the project repo can be found at `https://github.com/mandt-lab/RD-sandwich`.

A word on notation: this chapter follows the same information-theoretic notation as in Chapter 2.3, except that we denote the reproduction (random variable, point, alphabet) by $(Y, y, \mathcal{Y})$, instead of $(\hat{X}, \hat{x}, \hat{\mathcal{X}})$ to reduce visual clutter.

Rate-distortion (R-D) function, a key quantity in information theory, characterizes the fundamental limit of how much a data source can be compressed subject to a fidelity

criterion, by *any* compression algorithm. As researchers push for ever-improving compression performance, establishing the R-D function of a given data source is not only of scientific interest, but also reveals the possible room for improvement in compression algorithms. Previous work on this problem relied on distributional assumptions on the data source [Gibson, 2017] or only applied to discrete data. By contrast, this paper makes the first attempt at an algorithm for sandwiching the R-D function of a general (not necessarily discrete) source requiring only i.i.d. data samples. We estimate R-D sandwich bounds for a variety of artificial and real-world data sources, in settings far beyond the feasibility of any known method, and shed light on the optimality of neural data compression [Ballé et al., 2021, Yang et al., 2023a]. Our R-D upper bound on natural images indicates theoretical room for improving state-of-the-art image compression methods by at least one dB in PSNR at various bitrates.

6.1 Introduction

From storing astronomical images captured by the Hubble telescope, to delivering familiar faces and voices over video calls, data compression, i.e., communication of the "same" information but with less bits, is commonplace and indispensable to our digital life, and even arguably lies at the heart of intelligence [Mahoney, 2009]. While for lossless compression, there exist practical algorithms that can compress any discrete data arbitrarily close to the information theory limit [Ziv and Lempel, 1977, Witten et al., 1987], no such universal algorithm has been found for *lossy* data compression [Berger and Gibson, 1998], and significant research efforts have dedicated to lossy compression algorithms for various data. Recently, deep learning has shown promise for learning lossy compressors from raw data examples, with continually improving compression performance often matching or exceeding traditionally engineered methods [Minnen et al., 2018, Agustsson et al., 2020, Yang et al., 2021].

However, there are fundamental limits to the performance of any lossy compression algorithm, due to the inevitable trade-off between *rate*, the average number of bits needed to represent the data, and the *distortion* incurred by lossy representations. This trade-off is formally described by the rate-distortion (R-D) function, for a given *source* (i.e., the data distribution of interest; referred to as such in information theory) and distortion metric. The R-D function characterizes the best theoretically achievable rate-distortion performance by any compression algorithm, which can be seen as a lossy-compression counterpart and generalization of Shannon entropy in lossless compression.

Despite its fundamental importance, the R-D function is generally unknown analytically, and establishing it for general data sources, especially real world data, is a difficult problem [Gibson, 2017]. The default method for computing R-D functions, the Blahut-Arimoto algorithm [Blahut, 1972, Arimoto, 1972], only works for discrete data with a known probability mass function and has a complexity exponential in the data dimensionality. Applying it to an unknown data source requires discretization (if it is continuous) and estimating the source probabilities by a histogram, both of which introduce errors and are computationally infeasible beyond a couple of dimensions. Previous work characterizing the R-D function of images and videos [Hayes et al., 1970, Gibson, 2017] all assumed a statistical model of the source, making the results dependent on the modeling assumptions.

In this work, we make progress on this long-standing problem in information theory using tools from machine learning, and introduce new algorithms for upper and lower bounding the R-D function of a *general* (i.e., discrete, continuous, or neither), *unknown* memoryless source. More specifically,

1. Similarly to how a VAE with a discrete likelihood model minimizes an upper bound on the data entropy, we establish that *any β-VAE with a likelihood model induced by a distortion metric minimizes an upper bound on the data rate-distortion function.* We

thus open the deep generative modeling toolbox to the estimation of an upper bound on the R-D function.

2. We derive a lower bound estimator of the R-D function that can be made asymptotically exact and optimized by stochastic gradient ascent. Facing the difficulty of the problem involving global optimization, we restrict to a squared error distortion for a practical implementation.

3. We perform extensive experiments and obtain non-trivial sandwich bounds on various data sources, including GAN-generated artificial sources and real-world data from speech and physics. Our results shed light on the effectiveness of neural compression approaches [Ballé et al., 2021, Minnen et al., 2018, Minnen and Singh, 2020], and identify the *intrinsic* (rather than nominal) dimension of data as a key factor affecting the tightness of our lower bound.

4. Our estimated R-D upper bounds on high-resolution natural images (evaluated on the standard Kodak and Tecnick datasets) indicate theoretical room for improvement of state-of-the-art image compression methods by at least one dB in PSNR, at various bitrates.

We begin by reviewing the prerequisite rate-distortion theory in Section 6.2, then describe our upper and lower bound algorithms in Section 6.3 and Section 6.4, respectively. We discuss related work in Section 6.5, report experimental results in Section 6.6, and conclude in Section 6.7.

6.2 Background

Rate-distortion (R-D) theory deals with the fundamental trade-off between the average number of bits per sample (*rate*) used to represent a data source X and the *distortion*

incurred by the lossy representation Y. It asks the following question about the limit of lossy compression: for a given data source and a distortion metric (a.k.a., a fidelity criterion), what is the minimum number of bits (per sample) needed to represent the source at a tolerable level of distortion, regardless of the computation complexity of the compression procedure? The answer is given by the rate-distortion function $R(D)$. To introduce it, let the source and its reproduction take values in the sets $\mathcal{X}$ and $\mathcal{Y}$, conventionally called the *source* and *reproduction alphabets*, respectively. We define the data source formally by a random variable $X \in \mathcal{X}$ following a (usually unknown) distribution P_X, and assume a distortion metric $\rho : \mathcal{X} \times \mathcal{Y} \to [0, \infty)$ has been given, such as the squared error $\rho(x, y) = \|x - y\|^2$. The rate-distortion function is then defined by the following constrained optimization problem,

$$R(D) = \inf_{Q_{Y|X}:\ \mathbb{E}[\rho(X,Y)] \le D} I(X;Y), \tag{6.1}$$

where we consider all random transforms $Q_{Y|X}$ whose expected distortion is within the given threshold $D \ge 0$, and minimize the mutual information between the source X and its reproduced representation Y. [1] Shannon's lossy source coding theorems [Shannon, 1948, 1959] gave operational significance to the above mathematical definition of $R(D)$, as the minimum achievable rate with which any lossy compression algorithm can code i.i.d. data samples at a distortion level within D.

The R-D function thus gives the tightest lower bound on the rate-distortion performance of any lossy compression algorithm, and can inform the design and analysis of such algorithms. If the operational distortion-rate performance of an algorithm lies high above the source $R(D)$-curve $(D, R(D))$, then further performance improvement may be expected; otherwise, its performance is already close to theoretically optimal, and we may focus our attention on other aspects of the algorithm. As the R-D function does not have an analytical form in

[1] Both the expected distortion and mutual information terms are defined w.r.t. the joint distribution $P_X Q_{Y|X}$. We formally describe the general setting of the paper, including the technical definitions, in Appendix C.1.

general, we propose to estimate it from data samples, making the standard assumption that various expectations w.r.t. the true data distribution P_X exist and can be approximated by sample averages. When the source alphabet is finite, this assumption automatically holds, and $R(D)$ also provides a lower bound on the Shannon entropy of discrete data.

6.3 Upper Bound Algorithm

R-D theory [Cover and Thomas, 2006] tells us that every (distortion, rate) pair lying above the $R(D)$-curve is in theory realizable by a (possibly expensive) compression algorithm. An upper bound on $R(D)$ thus reveals what kind of compression performance is theoretically achievable. Towards this goal, we borrow the variational principle of the Blahut-Arimoto (BA) algorithm, but extend it to a general (e.g., non-discrete) source requiring only its samples. Our resulting algorithm optimizes a β-VAE whose likelihood model is specified by the distortion metric, a common case being a Gaussian likelihood with a fixed variance. For the first time, we establish this class of models as computing a model-agnostic upper bound on the source R-D function, as defined by a data compression task.

Variational Formulation. Following the BA algorithm [Blahut, 1972, Arimoto, 1972], we consider a Lagrangian relaxation of the constrained problem defining $R(D)$, which has the variational objective

$$\mathcal{L}(Q_{Y|X}, Q_Y, \lambda) := \mathbb{E}_{x\sim P_X}[KL(Q_{Y|X=x}\|Q_Y)] + \lambda \mathbb{E}_{P_X Q_{Y|X}}[\rho(X,Y)], \tag{6.2}$$

where Q_Y is a new, arbitrary probability measure on $\mathcal{Y}$. The first (*rate*) term is a variational upper bound on the mutual information $I(X;Y)$, and the second (*distortion*) term enforces the distortion tolerance constraint in Eq. 6.1. For each fixed $\lambda > 0$, the BA algorithm globally minimizes $\mathcal{L}$ w.r.t. the variational distributions $Q_{Y|X}$ and Q_Y by coordinate descent; at

convergence, the (distortion, rate) pair yields a point on the $R(D)$ curve [Csiszár, 1974b]. Unfortunately, the BA algorithm only applies when $\mathcal{X}$ and $\mathcal{Y}$ are finite (hence discrete), and the source distribution known. Otherwise, a preprocessing step is required to discretize a continuous source and/or estimate source probabilities by a histogram, which introduces a non-negligible bias. This bias, along with its exponential complexity in the data dimension, also makes BA infeasible beyond a couple of (usually 2 or 3) dimensions.

Proposed Method. To avoid these difficulties, we propose to apply (stochastic) gradient descent on $\mathcal{L}$ w.r.t. flexibly parameterized variational distributions $Q_{Y|X}$ and Q_Y. In this work we parameterize the distributions by neural networks, and predict the parameters of each $Q_{Y|X=x}$ by an *encoder* network $\phi(x)$ as in amortized inference [Kingma and Welling, 2014]. Given data samples, the estimates of rate and distortion terms of $\mathcal{L}$ yield a point that in expectation lies on an R-D upper bound $R_U(D)$, and we tighten this bound by optimizing $\mathcal{L}$; repeating this procedure for various λ traces out $R_U(D)$.

The objective $\mathcal{L}$ closely resembles the negative ELBO (NELBO) objective of a β-VAE [Higgins et al., 2017] if we view the reproduction space $\mathcal{Y}$ as the "latent space". The connection is immediate when $\mathcal{X}$ is continuous and a squared error ρ specifies the density of a Gaussian likelihood $p(x|y) \propto \exp(-\|x-y|^2)$. However, unlike in data compression, where $\mathcal{Y}$ is determined by the application (and often equal to $\mathcal{X}$ for a full-reference distortion), the latent space in a (β-)VAE typically has a lower dimension than $\mathcal{X}$, and a *decoder* network is used to parameterize a likelihood model in the data space. To capture this setup, we introduce a new, arbitrary latent space $\mathcal{Z}$ on which we define variational distributions $Q_{Z|X}, Q_Z$, and a (possibly stochastic) decoder function $\omega : \mathcal{Z} \to \mathcal{Y}$. This results in an extended objective, resembling a β-VAE with a likelihood density $p(x|z) \propto \exp\{-\rho(x, \omega(z)\}$,

$$J(Q_{Z|X}, Q_Z, \omega, \lambda) := \mathbb{E}_{x\sim P_X}[KL(Q_{Z|X=x}\|Q_Z)] + \lambda \mathbb{E}_{P_X Q_{Z|X}}[\rho(X, \omega(Z))]. \tag{6.3}$$

This objective is closely related to the original data compression task and provides an upper bound on the source $R(D)$, as follows. Treating $\mathcal{Z}$ as the reproduction alphabet, we can define a new distortion $\rho_\omega(x, z) := \rho(x, \omega(z))$, and a ω-induced R-D function, $R_\omega(D) := \inf_{Q_{Z|X}:\mathbb{E}[\rho_\omega(X,Z)]\leq D} I(X;Z)$, for each choice of ω. Our Theorem C.3.1 then guarantees that $R_\omega(D) \geq R(D)$, for any ω, and consequently the (distortion, rate) of J always lies above $R(D)$. Moreover, $R_\omega(D) = R(D)$ for a bijective ω, which offers some theoretical support for the use of invertible pixel-shuffle operations instead of upsampled convolutions in the decoder of image compression autoencoders [Theis et al., 2017a, Cheng et al., 2020]. We can now minimize the NELBO-like objective J w.r.t. the parameters of $(Q_{Z|X}, Q_Z, \omega)$ similar to training a β-VAE, knowing that we are optimizing an upper bound on the rate-distortion function of the source. This can be seen as the lossy counterpart to the lossless setting, where it is well-established that minimizing the NELBO minimizes an upper bound on the Shannon entropy of the source [Frey and Hinton, 1997, MacKay, 2003], the limit of lossless data compression.

The extended objective offers the freedom to define variational distributions on any suitable latent space $\mathcal{Z}$, rather than $\mathcal{Y}$, which we found to simplify the modeling task and yield tighter bounds. E.g., even if $\mathcal{Y}$ is high-dimensional and discrete, we can still work with densities on a continuous and lower-dimensional $\mathcal{Z}$ and draw upon tools such as normalizing flows [Kobyzev et al., 2021]. We can also treat Z as the concatenation of sub-vectors $[Z_1, Z_2, ..., Z_L]$, and parameterize Q_Z in terms of simpler component distributions $Q_Z = \prod_{l=1}^{L} Q_{Z_l|Z_{<l}}$ (similarly for $Q_{Z|X}$) as in a hierarchical VAE.

6.4 Lower Bound Algorithm

Without knowing the tightness of an R-D upper bound, we could be wasting time and resources trying to improve the R-D performance of a compression algorithm, when it is

in fact already close to the theoretical limit. This would be avoided if we could find a matching *lower* bound on $R(D)$. Unfortunately, the problem turns out to be much more difficult computationally. Indeed, every compression algorithm, or every pair of variational distributions $(Q_Y, Q_{Y|X})$ yields a point above $R(D)$. Conversely, establishing a lower bound requires disproving the existence of *any* compression algorithm that can conceivably operate below the $R(D)$ curve. In this section, we derive an algorithm that can in theory produce arbitrarily tight R-D lower bounds. However, as an indication of its difficulty, the problem requires *globally* maximizing a family of partition functions. By restricting to a continuous reproduction alphabet and a squared error distortion, we make some progress on this problem and obtain useful lower bounds especially on data with low intrinsic dimension (see Sec. 6.6).

Dual characterization of $R(D)$. While upper bounds on $R(D)$ arise naturally out of its definition as a minimization problem, a variational lower bound requires expressing $R(D)$ through a *maximization* problem. For this, we introduce a "dual" function as the optimum of the Lagrangian Eq. 6.2 (Q_Y is eliminated by replacing the rate upper bound with the exact mutual information $I(X;Y)$):

$$F(\lambda) := \inf_{Q_{Y|X}} I(X;Y) + \lambda \mathbb{E}[\rho(X,Y)]. \tag{6.4}$$

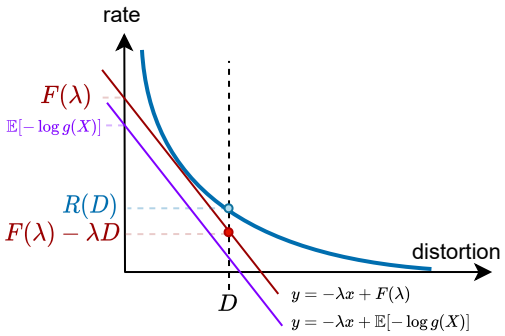


Figure 6.1: The geometry of the R-D lower bound problem. For a given slope $-\lambda$, we seek to maximize the *rate*-axis intercept, $\mathbb{E}[-\log g(X)]$, over all $g \geq 0$ functions admissible according to Eq. 6.6.

As illustrated by Fig. 6.1, $F(\lambda)$ is the maximum R-axis intercept of a straight line with slope $-\lambda$, among all such lines that lie below or tangent to $R(D)$; the R-D curve can then be found by taking the upper envelope of lines with slope $-\lambda$ and R-intercept $F(\lambda)$, i.e., $R(D) = \max_{\lambda \geq 0} F(\lambda) - \lambda D$. This key result is captured mathematically by Lemma C.1.1, and the following theorem:

Theorem 6.4.1. *[Csiszár, 1974c]* (As follows, all the expectations are with respect to the data source r.v. $X \sim P_X$.) *Under basic conditions (e.g., satisfied by a bounded ρ; see Appendix C.2.1), it holds that*

$$F(\lambda) = \max_{g(x)} \{\mathbb{E}[-\log g(X)]\}, \tag{6.5}$$

where the maximization is over all non-negative functions $g : \mathcal{X} \to [0, \infty)$ satisfying the constraint

$$\mathbb{E}\left[\frac{\exp(-\lambda\rho(X, y))}{g(X)}\right] = \int \frac{\exp(-\lambda\rho(x, y))}{g(x)} dP_X(x) \leq 1, \forall y \in \mathcal{Y}. \tag{6.6}$$

In other words, every admissible g yields a lower bound of $R(D)$, via an underestimator of the intercept $\mathbb{E}[-\log g(X)] \leq F(\lambda)$. We give the origin of this result in related work in Section 6.5.

Proposed Unconstrained Formulation. The constraint in Eq. 6.6 is concerning—it is a family of possibly infinitely many constraints, one for each y. To make the problem easier to work with, we propose to eliminate the constraints by the following transformation. Let g be defined in terms of another function $u(x) \geq 0$ and a scalar c depending on u, such that

$$g(x) := cu(x), \quad \text{where } c := \sup_{y \in \mathcal{Y}} \Psi_u(y), \quad \text{and } \Psi_u(y) := \mathbb{E}\left[\frac{\exp -\lambda\rho(X, y)}{u(X)}\right]. \tag{6.7}$$

This reparameterization of g is without loss of generality, and can be shown to always satisfy the constraint in Eq. 6.6. While this form of g bears a superficial resemblance to an energy-based model [LeCun et al., 2006], with $\frac{1}{c}$ resembling a normalizing constant, there is an important difference: $c = \sup_y \Psi_u(y)$ is in fact the supremum of a family of "partition functions" $\Psi_u(y)$ indexed by y; we thus refer to c as the *sup-partition function*. Although all

these quantities have λ-dependence, we omit this from our notation since λ is a fixed input parameter (as in the upper bound algorithm).

Consequently, F is now the result of *unconstrained* maximization over all u functions, and we obtain a lower bound on it by restricting u to a subset of functions with parameters θ (e.g., neural networks),

$$F(\lambda) = \max_{u \geq 0} \{\mathbb{E}[-\log u(X)] - \log \sup_{y \in \mathcal{Y}} \Psi_u(y)\} \geq \max_{\theta} \{\mathbb{E}[-\log u_\theta(X)] - \log \sup_{y \in \mathcal{Y}} \Psi_\theta(y)\}$$

Define the θ-parameterized objective $\ell(\theta) := \mathbb{E}[-\log u_\theta(X)] - \log c(\theta)$, with $c(\theta) = \sup_{y \in \mathcal{Y}} \Psi_\theta(y)$. Given samples of X, we can in principle maximize $\ell(\theta)$ by (stochastic) gradient ascent. However, computing the sup-partition function $c(\theta)$ poses serious computation challenges: even evaluating $\Psi_\theta(y)$ for a single y value involves a potentially high-dimensional integral w.r.t. P_X; this is only exacerbated by the need to globally optimize w.r.t. y, an NP-hard problem even in one-dimension.

Proposed Method. To tackle this problem, we propose an over-estimator of the sup-partition function inspired by IWAE [Burda et al., 2015]. Fix θ for now; we denote the integrand in Eq. 6.7 by $\psi(x, y) := \frac{\exp -\lambda \rho(x,y)}{u(x)}$ (so $c = \sup_{y \in \mathcal{Y}} \mathbb{E}[\psi(X, y)]$), and omit the dependence on θ to simplify notation. Given $k \geq 1$ i.i.d. random variables $X_1, ..., X_k \sim P_X$, define the estimator $C_k := \sup_y \frac{1}{k} \sum_i \psi(X_i, y)$. We prove in Theorem C.3.2 that $\mathbb{E}[C_1] \geq \mathbb{E}[C_2] \geq ... \geq c$, i.e., C_k is in expectation an over-estimator of the sup-partition function c. Similarly to the Importance-Weighted ELBO [Burda et al., 2015], the bias of this estimator decreases monotonically as $k \to \infty$, and asymptotically vanishes under regularity assumptions. In light of this, we replace c by $\mathbb{E}[C_k]$ and obtain a k-sample under-estimator of the objective $\ell(\theta)$ (which in turn underestimates $F(\lambda)$):

$$\ell_k(\theta) := \mathbb{E}[-\log u_\theta(X)] - \log \mathbb{E}[C_k]; \quad \text{moreover, } \ell_1(\theta) \leq \ell_2(\theta) \leq ... \leq \ell(\theta).$$

In order to apply stochastic gradient ascent, we overcome two more technical hurdles. First, each draw of C_k requires solving a global maximization problem. We note that by restricting to a squared-error ρ and $\mathcal{Y} = \mathcal{X}$, C_k can be computed by finding the mode of a Gaussian mixture density; for this we use the method of Carreira-Perpinan [2000], essentially by hill-climbing from each of the k centroids. Second, to turn $-\log \mathbb{E}[C_k]$ into an expectation, we follow Poole et al. [2019] and underestimate it by linearizing $-\log$ around a scalar parameter $\alpha > 0$, resulting in the following lower bound objective:

$$\tilde{\ell}_k(\theta) := \mathbb{E}[-\log u_\theta(X)] - \mathbb{E}[C_k]/\alpha - \log \alpha + 1. \tag{6.8}$$

$\tilde{\ell}_k(\theta)$ can finally be estimated by sample averages, and yields a lower bound on the optimal intercept $F(\lambda)$, via $\tilde{\ell}_k(\theta) \leq \ell_k(\theta) \leq \ell(\theta) \leq F(\lambda)$. A trained model u_{θ^*} then yields an R-D lower bound, $R_L(D) = -\lambda D + \tilde{\ell}_k(\theta^*)$. We give a more detailed derivation and pseudocode in Appendix C.4.

6.5 Related Work

Machine Learning and Compression: The past few years have seen significant progress in applying machine learning to lossy data compression. Theis et al. [2017a], Ballé et al. [2017] first showed that a particular type of β-VAE can be trained to perform data compression using the same objective as Eq. 6.3. The variational distributions in such a model have shape restrictions to simulate quantization and entropy coding [Ballé et al., 2017]. Our upper bound is directly inspired by this line of work, and suggests that such a model can in principle compute the source R-D function when equipped with sufficiently expressive variational distributions and a "rich enough" decoder (see Sec. 6.3). We note however not all compressive autoencoders admit a probabilistic formulation [Theis et al., 2017a]; recent work has found training with hard quantization to improve compression performance [Minnen

and Singh, 2020], and methods have been developed [Agustsson and Theis, 2020, Yang et al., 2020a] to reduce the gap between approximate quantization at training time and hard quantization at test time. Departing from compressive autoencoders, Yang et al. [2020b] and Flamich et al. [2020b] use Gaussian β-VAEs for data compression and exploit the flexibility of variable-width Gaussian posteriors. Flamich et al. [2020b]'s *relative entropy coding* method, and the related *reverse channel coding* or *channel simulation* algorithms [Theis and Yosri, 2021] (reviewed in Section 3.2.2 of the current thesis), can transmit a sample of $Q_{Z|X}$ with a rate close to that optimized by our upper bound model in Eq. 6.3, more precisely, $I(X;Z) + \log(I(X;Z)+1) + O(1)$. i.e., in this one-shot setting (which is standard for neural image compression), $R(D)$ is no longer achievable; rather, the achievable R-D performance is characterized by $R(D) + \log(R(D)+1) + O(1)$. Therefore, our R-D bounds can be shifted upwards by this logarithmic factor to give an estimate of the achievable R-D performance in this setting.

Information theory has also broadly influenced unsupervised learning and representation learning. The Information Bottleneck method [Tishby et al., 2000] was directly motivated by, and extends R-D theory and the BA algorithm. Alemi et al. [2018] analyzed the relation between generative modeling and representation learning with a similar R-D Lagrangian to Eq. 6.2, but used an abstract, model-dependent distortion $\rho(y,x) := -\log p(x|y)$ with an arbitrary $\mathcal{Y}$ and without considering a data compression task. Recently, Huang et al. [2020] proposed to evaluate decoder-based generative models by computing a restricted version of $R_\omega(D)$ (with Q_Y fixed); our result in Sec. 6.3 ($R_\omega(D) \geq R(D)$) gives a principled way to interpret and compare these model-dependent R-D curves.

Information Theory: While the BA algorithm [Blahut, 1972, Arimoto, 1972] computes the $R(D)$ of a discrete source with a known distribution, no tool currently exists for the general and unknown case. Riegler et al. [2018] share our goal of computing $R(D)$ of a

general source, but still require the source to be known analytically and supported on a known reference measure. Harrison and Kontoyiannis [2008] consider the same setup as ours of estimating $R(D)$ of an unknown source from samples, but focus on purely theoretical aspects, assuming prefect optimization. They prove statistical consistency of such estimators for a general class of alphabets and distortion metrics, assuring that our stochastic bounds on $R(D)$ optimized from data samples, when given unlimited computation and samples, can converge to the true $R(D)$. Perhaps closest in spirit to our work is by Gibson [2017], who estimates lower bounds on $R(D)$ of speech and video using Gaussian autoregressive models of the source. However, the correctness of the resulting bounds depends on the modeling assumptions.

A variational lower bound on $R(D)$ was already proposed by Shannon [1959], and later extended [Berger, 1971] to the present version similar to Theorems 6.4.1 and C.2.1. In the finite-alphabet case, the maximization characterization of $R(D)$ follows from taking the Lagrange dual of its standard definition in Eq. 6.1; the dual problem can then be solved by convex optimization [Chiang and Boyd, 2004], but faces the same computational difficulties as the BA algorithm. Csiszár [1974c] proved the general result in Theorem 6.4.1, applicable to abstract alphabets (in particular, with source X taking values in an arbitrary probability space), by analyzing the fixed-point conditions of the BA algorithm.

6.6 Experiments

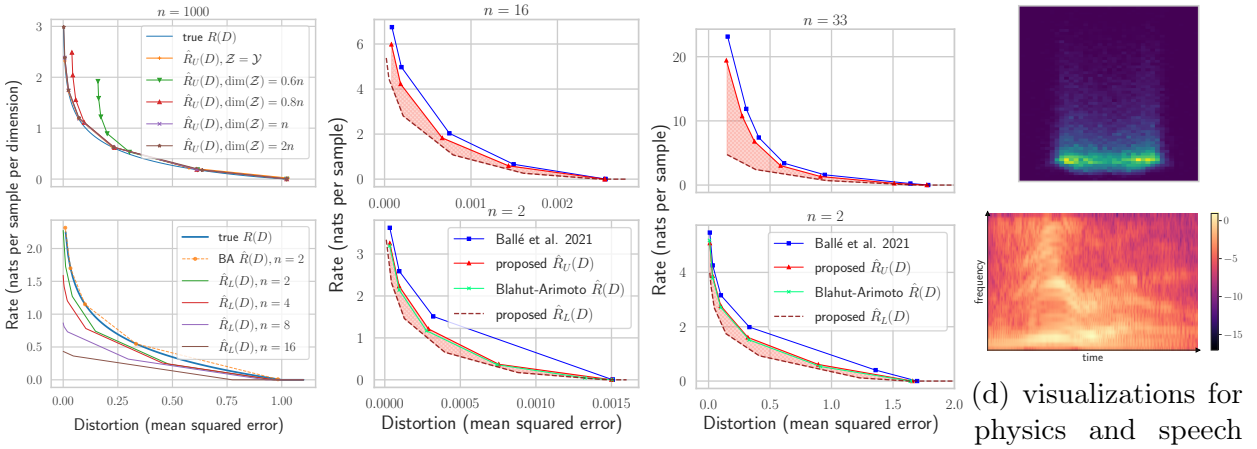
We estimate the R-D functions of a variety of artificial and real-world data sources, in settings where the BA algorithm is infeasible and no prior known method has been applied. On **Gaussian sources**, our upper bound algorithm is shown to converge to the *exact* R-D function, while our lower bounds become increasingly loose in higher dimensions, an issue we investigate subsequently. We obtain tighter sandwich bounds on **particle physics** and

speech data than on similar dimensional Gaussians, and compare with the performance of neural compression. We further investigate looseness in the lower bound, experimentally establishing the *intrinsic dimension* of the data as a much more critical contributing factor than the nominal/ambient dimension. Indeed, we obtain tight sandwich bounds on **high-dimension GAN-generated images** with a low intrinsic dimension, and compare with popular neural image compression methods. Finally, we estimate bounds on the R-D function of **natural images**. The intrinsic dimension is likely too high for our lower bound to be useful, while our upper bounds on the Kodak and Tecnick datasets imply at least one dB (in PSNR) of theoretical room for improving state-of-the-art image compression methods, at various bitrates. We also validate our bounds against the BA algorithm over the various data sources, using a 2D marginal of the source to make BA feasible. We provide experimental details and additional results in Appendix C.5 and C.6.

6.6.1 Gaussian Sources

We start by applying our algorithms to the factorized Gaussian distribution, one of the few sources with an analytical R-D function. We randomly generate the Gaussian sources in increasing dimensions.

For the upper bound algorithm, we let Q_Y and $Q_{Y|X}$ be factorized Gaussians with learned parameters, predicting the parameters of $Q_{Y|X}$ by a 1-layer MLP encoder. As shown in Fig. 6.2a-*top*, on a $n = 1000$ dimensional Gaussian (the results are similar across all the n we tested), our upper bound (**yellow** curve) accurately recovers the analytical $R(D)$. We also optimized the variational distributions in a latent space $\mathcal{Z}$ with varying dimensions, using an MLP decoder to map from $\mathcal{Z}$ to $\mathcal{Y}$ (see Sec. 6.3). The resulting bounds are similarly tight when the latent dimension matches or exceeds the data dimension n (**green**, **brown**), but become loose otherwise (**red** and **purple** curves), demonstrating the importance of a rich



(a) Gaussian (b) particle physics (c) speech (d) visualizations for physics and speech data

Figure 6.2: **6.2a** *top*: R-D upper bound estimates on a randomly generated n =1000-dimensional Gaussian source; *bottom*: R-D lower bound estimates on standard Gaussians with increasing dimensions (the result of the BA algorithm for $n = 2$ is also shown for reference). **6.2b** *top*: estimated R-D bounds and the R-D performance of Ballé et al. [2021] on the particle physics dataset; *bottom*: the same experiment but on a 2D marginal distribution of the data, to compare with the BA algorithm. **6.2c**: the same set of experiments as in **6.2b**, repeated on the speech dataset. **6.2d** *top*: histogram of the 2D marginal distribution of the physics data; *bottom*: example spectrogram computed on a speech clip.

enough latent space for a tight R-D bound, as suggested by our Theorem C.3.1.

For the lower bound algorithm, we parameterize $\log u$ by a 2-layer MLP, and study the effect of source dimension n and the number of samples k used in our estimator C_k (and objective $\tilde{\ell}_k$). To simplify comparison of results across different source dimensions, here we consider standard Gaussian sources, whose R-D curve does not vary with n if we scale the rate by $\frac{1}{n}$ (i.e., rate per sample per dimension); the results on randomly generated Gaussians are similar. First, we fix $k = 1024$; Fig. 6.2a-*bottom* shows that the resulting bounds quickly become loose with increasing source dimension. This is due to the bias of our estimator C_k for the sup-partition function, which causes under-estimation in the objective $\tilde{\ell}_k$. While C_k is defined similarly to an M-estimator [Van der Vaart, 2000], analyzing its convergence behavior is not straightforward, as it depends on the function u being learned. In this experiment, we observe the bias of C_k is amplified by an increase in n or λ, such that an increasingly large k is required for effective training. In another experiment, we estimate that the k needed to

close the gap in the lower bound increases exponentially in n; see results in Fig. C.1, and a detailed discussion on this, in Appendix C.5.2. Fortunately, as we see in Sec. 6.6.3, the bias in our lower bound appears to depend on the *intrinsic* rather than (often much higher) nominal dimension of data, giving us a more favorable trade-off between computation and a tighter lower bound as controlled by k.

6.6.2 Data from Particle Physics and Speech

The quickly deteriorating lower bound on higher (even 16) dimensional Gaussians may seem disheartening. However, the Gaussian is also the hardest continuous source to compress under squared error [Gibson, 2017], and real-world data often exhibits considerably more structure than Gaussian noise. In this subsection, we experiment on data from particle physics [Howard et al., 2021] and speech [Jackson et al., 2018], and indeed obtain improved sandwich bounds compared to Gaussians.

First, we consider the Z-boson decay dataset from Howard et al. [2021], containing n=16-dimensional vectors of four-momenta information from independent particle decay events. We ran the neural compression method from [Ballé et al., 2021], as well as our bounding algorithms with similar configurations to before, except we fix $k = 2048$ for the lower bound, and use a normalizing flow for Q_Z in the upper bound model for better expressiveness. Fig. 6.2b *top* shows the resulting estimated R-D bounds (sandwiched region colored in **red**) and the operational R-D curve for Ballé et al. [2021] (**blue**). The resulting sandwich bounds appear significantly tighter here than on the Gaussian source with equal dimension ($n = 16$, bottom curve in Fig. 6.2a *bottom*), and the neural compression method operates with a relatively small average overhead of 0.5 nat/sample relative to our upper bound. To also compare to the ground-truth $R(D)$ as estimated by the BA algorithm, we created a 2-dimensional marginal data distribution (plotted in Fig. 6.2d *top*), so that the BA algorithm can be feasibly run

with a fine discretization grid. As shown in Fig. 6.2b, the BA estimate of $R(D)$ (**green**) almost overlaps with our upper bound on the 2D marginal, and is tightly sandwiched from below by our lower bound.

We then repeat the same experiments on speech data from the Free Spoken Digit Dataset [Jackson et al., 2018]. We constructed our dataset by pre-processing the audio recordings into spectrograms (see, e.g., Fig. 6.2d *bottom*), then treating the resulting $n = 33$-dimensional frequency feature vectors as independent across time. As shown in Fig. 6.2c, the gap in our R-D bounds appears wider than on the physics dataset (*top*), but the results on the corresponding 2D marginal appear similar (*bottom*).

6.6.3 The Effect of Intrinsic v.s. Nominal Dimension of Data

It is known that learning a manifold has a complexity that depends on the intrinsic dimension of the manifold, but not on its ambient dimension [Narayanan and Mitter, 2010]. Our experiments suggest a similar phenomenon for our lower bound, and show that we can still obtain tight sandwich bounds on data with a sufficiently low *intrinsic* dimension despite the high ambient dimension.

First, we explore the effect of increasing the ambient dimension, while keeping the intrinsic dimension of the data fixed. We borrow the 2D banana-source from Ballé et al. [2021] (visualized in Fig. 6.3), and randomly embed it in $\mathbb{R}^n$. As shown in Fig. 6.4 and Fig. 6.5, our sandwich bounds on the 2D source appear tight, and closely agree with BA (similar to the results in Fig. 6.2); moreover, the tightness appears unaffected by the increase in ambient dimension n to 4, 16, and 100 (we verified this for n up to 1000). Unlike in the Gaussian experiment, where increasing n required seemingly exponentially larger k for a good lower bound, here a constant $k = 1024$ worked well for all n.

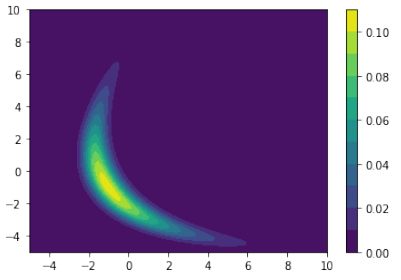


Figure 6.3: Density plot of the 2D banana-shaped distribution from Ballé et al. [2021]. The distribution is obtained by a nonlinear transform of a 2D Gaussian.

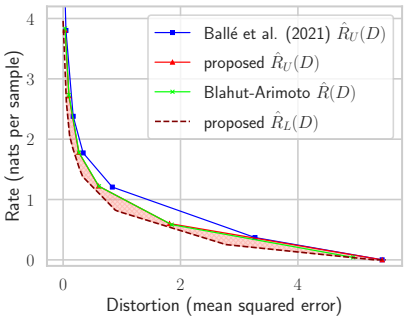


Figure 6.4: R-D sandwich bounds and the R-D performance of non-linear transform coding [Ballé et al., 2021] on the banana-shaped source. For reference, the BA algorithm is also run on a fine discretization of the source.

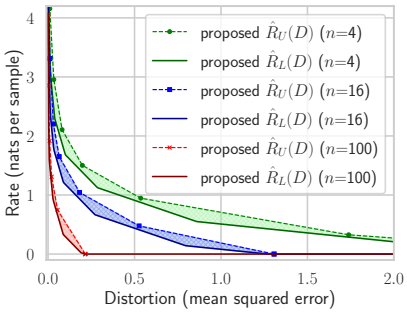


Figure 6.5: R-D sandwich bounds on higher-dimension embeddings of the banana-shaped source. The tightness of our bounds appear unaffected by the increasing n.

Next, we experiment on high-dimension GAN-generated images with varying intrinsic dimension, and obtain R-D sandwich bounds that help assess neural image compression methods. Following Pope et al. [2021], we generate 128×128 images of `basenji` from a pre-trained GAN, and control the intrinsic dimension by zeroing out all except d dimensions of the noise input to the GAN. As shown in Fig. 6.6-Left, the images appear highly realistic, showing dogs with subtly different body features and in various poses. We implemented a 6-layer ResNet-VAE [Kingma et al., 2016] for our upper bound model, and a simple convolutional network for our lower bound model. Fig. 6.6-Middle plots our resulting R-D bounds and the sandwiched region (**red**), along with the operational R-D curves (**blue, green**) of neural image compression methods [Minnen et al., 2018, Minnen and Singh, 2020] trained on the GAN images, for $d = 2$ and $d = 4$. We see that despite the high dimensionality ($n = 128 \times 128 \times 3$), the images require few nats to compress; e.g., for $d = 4$, we estimate $R(D)$ to be between ~ 4 and 8 nats per sample at $D = 0.001$ (30 dB in PSNR). Notably, the R-D curve of Minnen and Singh [2020] stays roughly within a factor of 2 above our estimated true $R(D)$ region. The neural compression methods show improved performance as d decreases from 4 to 2, following the same trend as our R-D bounds. This demonstrates the effectiveness of

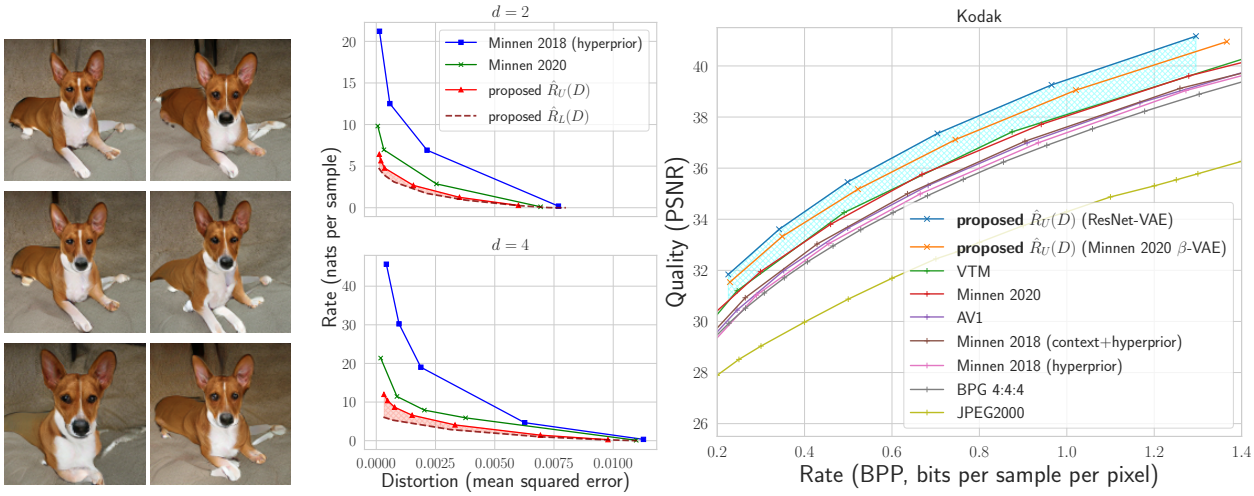


Figure 6.6: **Left**: 128×128 GAN-generated images, with intrinsic dimension $d = 4$. **Middle**: Bounds on $R(D)$ of GAN images, for $d = 2$ (top) and $d = 4$ (bottom). **Right**: Quality-rate curves of ours and state-of-the-art image compression methods on Kodak [1993], corresponding to R-D upper bounds.

learned compression methods at adapting to low-dimensional structure in data, in contrast to traditional methods such as JPEG, whose R-D curve on this data does not appear to vary with d and lies orders of magnitude higher.

6.6.4 Natural Images

To establish upper bounds on the R-D function of natural images, we define variational distributions $(Q_Z, Q_{Z|X})$ on a Euclidean latent space for simplicity, and parameterize them as well as a learned decoder ω via hierarchical VAEs. We consider two VAE architectures: 1. we borrow the convolutional autoencoder architecture of a state-of-the-art image compression model [Minnen and Singh, 2020], but use factorized Gaussians for the variational distributions (we still keep the deep factorized hyperprior, but no longer convolve it with a uniform prior); 2. we also reuse our ResNet-VAE architecture with 6 stochastic layers from the GAN experiments (Sec. 6.6.3). We trained the models with mean-squared error (MSE) distortion and various λ on the COCO 2017 [Lin et al., 2014] images, and evaluated them on the Kodak [1993] and Tecnick [Asuni and Giachetti, 2014] datasets. Following image compression conventions, we

report the rate in bits-per-pixel, and the quality (i.e., negative distortion) in PSNR averaged over the images for each (λ, model) pair.[2] The resulting *quality-rate* (Q-R) curves can be interpreted as giving upper bounds on the R-D functions of the image-generating distributions. We plot them in Fig. 6.6, along with the Q-R performance (in actual bitrate) of various traditional and learned image compression methods [Ballé et al., 2017, Minnen et al., 2018, Minnen and Singh, 2020], for the Kodak dataset (see similar results on Tecnick in Appendix Fig. C.5). Our β-VAE version of [Minnen and Singh, 2020] (**orange**) lies on average 0.7 dB higher than the Q-R curves of the original compression model (**red**) and VTM (**green**). With a deeper latent hierarchy, our ResNet-(β-)VAE gives a higher Q-R curve (**blue**) that is on average 1.1 dB above the state-of-the-art Q-R curves (gap shaded in **cyan**). We leave it to future work to investigate which choice of autoencoder architecture and variational distributions are most effective, as well as how the theoretical R-D performance of such a β-VAE can be realized by a practical compression algorithm (see discussions in Sec. 6.5).

6.7 Discussions

In this work, we proposed machine learning techniques to computationally bound the rate-distortion function of a data source, a key quantity that characterizes the fundamental performance limit of all lossy compression algorithms, but is largely unknown. Departing from prior work in the information theory community [Gibson, 2017, Riegler et al., 2018], our approach applies broadly to general data sources and requires only i.i.d. samples, making it more suitable for real-world application.

Our upper bound method is a gradient descent version of the classic Blahut-Arimoto algorithm, and closely relates to (and extends) variational autoencoders from neural lossy compression

[2]Technically, to estimate an R-D upper bound with MSE as ρ, one should compute the distortion by averaging MSE (instead of PSNR) on images; however, the results of many image compression baselines are only available in average PSNR.

research. Our lower bound method optimizes a dual characterization of the R-D function, which has been known for some time but seen little application outside of theoretical work. Due to difficulties involving global optimization, our lower bound currently requires a squared error distortion for tractability in the continuous case, and is only tight on data sources with a low *intrinsic* dimension. We hope that a better understanding of the lower bound problem will lead to improved algorithms in the future.

To properly interpret bounds on the R-D function, we emphasize that the significance of the R-D function is two-fold: 1. for a given distortion tolerance D, no coding procedure can operate with a rate less than $R(D)$, and that 2. this rate is *asymptotically* achievable by a potentially expensive block code. Thus, while a lower bound makes a universal statement about what performance is "too good to be true", the story is more subtle for the upper bound. The achievability proof relies on a random coding procedure that jointly compresses multiple data samples in arbitrarily long blocks [Shannon, 1959]. When compressing at a finite block length b, $R(D)$ is generally no longer achievable due to a $O(\frac{1}{\sqrt{b}})$ rate overhead [Kontoyiannis, 2000, Kostina and Verdú, 2012]. Extending our work to the settings of finite block lengths or non-memoryless sources may be additional future directions.

Chapter 7

Statistical and Optimal-Transport Perspectives on the Estimation of the Rate-Distortion Function

Most of the material in this chapter has been published in the following conference paper,

Estimating the Rate-Distortion Function by Wasserstein Gradient Descent. Yibo Yang, Stephan Eckstein, Marcel Nutz, Stephan Mandt. *Conference on Neural Information Processing Systems (NeurIPS), 2023.*

Additional results and supplementary information can be found in Appendix D, and the project repo can be found at `https://github.com/yiboyang/wgd`.

A word on notation: this chapter follows the same information-theoretic notation as in Chapter 6, and introduces new notation from optimal transport.

In the theory of lossy compression, the rate-distortion (R-D) function $R(D)$ describes how much a data source can be compressed (in bit-rate) at any given level of fidelity (distortion). Obtaining $R(D)$ for a given data source establishes the fundamental performance limit for all compression algorithms. We propose a new method to estimate $R(D)$ from the perspective of optimal transport. Unlike the classic Blahut–Arimoto algorithm which fixes the support of the reproduction distribution in advance, our Wasserstein gradient descent algorithm learns the support of the optimal reproduction distribution by moving particles. We prove its local convergence and analyze the sample complexity of our R-D estimator based on a connection to entropic optimal transport. Experimentally, we obtain comparable or tighter bounds than state-of-the-art neural network methods on low-rate sources while requiring considerably less tuning and computation effort. We also highlight a connection to maximum-likelihood deconvolution and introduce a new class of sources that can be used as test cases with known solutions to the R-D problem.

7.1 Introduction

The rate-distortion (R-D) function $R(D)$ occupies a central place in the theory of lossy compression. For a given data source and a fidelity (or *distortion*) criterion, $R(D)$ characterizes the minimum possible communication cost needed to reproduce a source sample within an error threshold of D, by *any* compression algorithm [Shannon, 1959]. A basic scientific and practical question is therefore establishing $R(D)$ for any given data source of interest, which helps assess the (sub)optimality of the compression algorithms and guide their development. The classic algorithm by Blahut [1972] and Arimoto [1972] assumes a known discrete source and computes its $R(D)$ by an exhaustive optimization procedure. This often has limited applicability in practice, and a line of research has sought to instead *estimate* $R(D)$ from data samples [Harrison and Kontoyiannis, 2008, Gibson, 2017], with recent methods [Yang

and Mandt, 2022, Lei et al., 2023a] inspired by deep generative models.

In this work, we propose a new approach to R-D estimation from the perspective of optimal transport. Our starting point is the formulation of the R-D problem as the minimization of a certain rate functional [Harrison and Kontoyiannis, 2008] over the space of probability measures on the reproduction alphabet. Optimization over such an infinite-dimensional space has long been studied under gradient flows [Ambrosio et al., 2008], and we consider a concrete algorithmic implementation based on moving particles in space. This formulation of the R-D problem also suggests connections to entropic optimal transport and non-parametric statistics, each offering us new insight into the solution of the R-D problem under a quadratic distortion. More specifically, our contributions are three-fold:

First, we introduce a neural-network-free $R(D)$ upper bound estimator for continuous alphabets. We implement the estimator by Wasserstein gradient descent (WGD) over the space of reproduction distributions. Experimentally, we found the method to converge much more quickly than state-of-the-art neural methods with hand-tuned architectures, while offering comparable or tighter bounds.

Second, we theoretically characterize convergence of our WGD algorithm and the sample complexity of our estimator. The latter draws on a connection between the R-D problem and that of minimizing an entropic optimal transport (EOT) cost relative to the source measure, allowing us to turn statistical bounds for EOT [Mena and Niles-Weed, 2019] into finite-sample bounds for R-D estimation.

Finally, we introduce a new, rich class of sources with known ground truth, including Gaussian mixtures, as a benchmark for algorithms. While the literature relies on the Gaussian or Bernoulli for this purpose, we use the connection with maximum likelihood deconvolution to show that a Gaussian convolution of *any* distribution can serve as a source with a known solution to the R-D problem.

7.2 Lossy Compression, Entropic Optimal Transport, and MLE

This section introduces the R-D problem and its rich connections to entropic optimal transport and statistics, along with new insights into its solution. Sec. 7.2.1 sets the stage for our method (Sec. 7.4) by a known formulation of the standard R-D problem as an optimization problem over a space of probability measures. Sec. 7.2.2 discusses the equivalence between the R-D problem and a projection of the source distribution under entropic optimal transport; this is a key to our sample complexity results in Sec. 7.4.3. Lastly, Sec 2.3 gives a statistical interpretation of R-D as maximum-likelihood deconvolution and uses it to analytically derive a segment of the R-D curve for a new class of sources under quadratic distortion; this allows us to assess the optimality of algorithms in experiment Sec. 7.5.1.

7.2.1 Setup

For a memoryless data source X with distribution P_X, its rate-distortion (R-D) function describes the minimum possible number of bits per sample needed to reproduce the source within a prescribed distortion threshold D. Let the source and reproduction take values in two sets $\mathcal{X}$ and $\mathcal{Y}$, known as the source and reproduction *alphabets*, and let $\rho : (\mathcal{X}, \mathcal{Y}) \to [0, \infty)$ be a given distortion function. The R-D function is defined by the following optimization problem [Polyanskiy and Wu, 2022],

$$R(D) = \inf_{Q_{Y|X}:\mathbb{E}_{P_X Q_{Y|X}}[\rho(X,Y)]\leq D} I(X;Y), \tag{7.1}$$

where $Q_{Y|X}$ is any Markov kernel from $\mathcal{X}$ to $\mathcal{Y}$ conceptually associated with a (possibly) stochastic compression algorithm, and $I(X;Y)$ is the mutual information of the joint distribution $P_X Q_{Y|X}$.

For ease of presentation, we now switch to a more abstract notation without reference to random variables. We provide the precise definitions in Appendix D. Let $\mathcal{X}$ and $\mathcal{Y}$ be standard Borel spaces; let $\mu \in \mathcal{P}(\mathcal{X})$ be a fixed probability measure on $\mathcal{X}$, which should be thought of as the source distribution P_X. For a measure π on the product space $\mathcal{X} \times \mathcal{Y}$, the notation π_1 (or π_2) denotes the first (or second) marginal of π. For any $\nu \in \mathcal{P}(\mathcal{Y})$, we denote by $\Pi(\mu, \nu)$ the set of couplings between μ and ν (i.e., $\pi_1 = \mu$ and $\pi_2 = \nu$). Similarly, $\Pi(\mu, \cdot)$ denotes the set of measures π with $\pi_1 = \mu$. Throughout the paper, K denotes a transition kernel (conditional distribution) from $\mathcal{X}$ to $\mathcal{Y}$, and $\mu \otimes K$ denotes the product measure formed by μ and K. Then $R(D)$ is equivalent to

$$R(D) = \inf_{K: \int \rho d(\mu \otimes K) \le D} H(\mu \otimes K | \mu \otimes (\mu \otimes K)_2) = \inf_{\pi \in \Pi(\mu, \cdot): \int \rho d\pi \le D} H(\pi | \pi_1 \otimes \pi_2), \tag{7.2}$$

where H denotes relative entropy, i.e., for two measures α, β defined on a common measurable space, $H(\alpha|\beta) := \int \log(\frac{d\alpha}{d\beta}) d\alpha$ when α is absolutely continuous w.r.t β, and infinite otherwise.

To make the problem more tractable, we follow the approach of the classic Blahut–Arimoto algorithm [Blahut, 1972, Arimoto, 1972] (to be discussed in Sec. 7.3.1) and work with an equivalent unconstrained Lagrangian problem as follows. Instead of parameterizing the R-D function via a distortion threshold D, we parameterize it via a Lagrange multiplier $\lambda \ge 0$. For each fixed λ (usually selected from a predefined grid), we aim to solve the following optimization problem,

$$F_\lambda(\mu) := \inf_{\nu \in \mathcal{P}(\mathcal{Y})} \inf_{\pi \in \Pi(\mu, \cdot)} \lambda \int \rho d\pi + H(\pi | \mu \otimes \nu). \tag{7.3}$$

Geometrically, $F_\lambda(\mu) \in \mathbb{R}$ is the y-axis intercept of a tangent line to the $R(D)$ with slope $-\lambda$, and $R(D)$ is determined by the convex envelope of all such tangent lines [Gray, 2011]. To simplify notation, we often drop the dependence on λ (e.g., we write $F(\mu) = F_\lambda(\mu)$) whenever it is harmless.

To set the stage for our later developments, we write the unconstrained R-D problem as

$$F_\lambda(\mu) = \inf_{\nu \in \mathcal{P}(\mathcal{Y})} \mathcal{L}_{BA}(\mu, \nu), \tag{7.4}$$

$$\mathcal{L}_{BA}(\mu, \nu) := \inf_{\pi \in \Pi(\mu, \cdot)} \lambda \int \rho d\pi + H(\pi | \mu \otimes \nu) = \inf_{K} \lambda \int \rho d(\mu \otimes K) + H(\mu \otimes K | \mu \otimes \nu), \tag{7.5}$$

where we refer to the optimization objective $\mathcal{L}_{BA}$ as the *rate function* [Harrison and Kontoyiannis, 2008]. We abuse the notation to write $\mathcal{L}_{BA}(\nu) := \mathcal{L}_{BA}(\mu, \nu)$ when it is viewed as a function of ν only, and refer to it as the *rate functional*. The rate function characterizes a generalized Asymptotic Equipartition Property, where $\mathcal{L}_{BA}(\mu, \nu)$ is the asymptotically optimal cost of lossy compression of data $X \sim \mu$ using a random codebook constructed from samples of ν [Dembo and Kontoyiannis, 2002]. Notably, the optimization in (7.5) can be solved analytically [Csiszár, 1974a, Lemma 1.3], and $\mathcal{L}_{BA}$ simplifies to

$$\mathcal{L}_{BA}(\mu, \nu) = \int_{\mathcal{X}} -\log \left(\int_{\mathcal{Y}} e^{-\lambda \rho(x,y)} \nu(dy) \right) \mu(dx). \tag{7.6}$$

In practice, the source μ is only accessible via independent samples, on the basis of which we propose to estimate its $R(D)$, or equivalently $F(\mu)$. Let μ^m denote an m-sample empirical measure of μ, i.e., $\mu^m = \sum_{i=1}^m \delta_{x_i}$ with $x_{1,\dots,n}$ being independent samples from μ, which should be thought of as the "training data". Following Harrison and Kontoyiannis [2008], we consider two kinds of (plug-in) estimators for $F(\mu)$: (1) the non-parametric estimator $F(\mu^m)$, and (2) the parametric estimator $F^{\mathcal{H}}(\mu^m) := \inf_{\nu \in \mathcal{H}} \mathcal{L}_{BA}(\mu^m, \nu)$, where $\mathcal{H}$ is a family of probability measures on $\mathcal{Y}$. Harrison and Kontoyiannis [2008] showed that under rather broad conditions, both kinds of estimators are strongly consistent, i.e., $F(\mu^m)$ converges to $F(\mu)$ (and respectively, $F^{\mathcal{H}}(\mu^m)$ to $F^{\mathcal{H}}(\mu)$) with probability one as $m \to \infty$. Our algorithm will implement the parametric estimator $F^{\mathcal{H}}(\mu^m)$ with $\mathcal{H}$ chosen to be the set of probability measures with finite support, and we will develop finite-sample convergence results for both

kinds of estimators in the continuous setting (Proposition. 7.4.3).

7.2.2 Connection to Entropic Optimal Transport

The R-D problem turns out to have a close connection to entropic optimal transport (EOT) [Peyré and Cuturi, 2019], which we will exploit in Sec. 7.4.3 to obtain sample complexity results under our approach. For $\epsilon > 0$, the entropy-regularized optimal transport problem is given by

$$\mathcal{L}_{EOT}(\mu, \nu) := \inf_{\pi \in \Pi(\mu,\nu)} \int \rho d\pi + \epsilon H(\pi | \mu \otimes \nu). \tag{7.7}$$

We now consider the problem of projecting μ onto $\mathcal{P}(\mathcal{Y})$ under the cost $\mathcal{L}_{EOT}$:

$$\inf_{\nu \in \mathcal{P}(\mathcal{Y})} \mathcal{L}_{EOT}(\mu, \nu). \tag{7.8}$$

In the OT literature this is known as the (regularized) Kantorovich estimator [Bassetti et al., 2006] for μ, and can also be viewed as a Wasserstein barycenter problem [Agueh and Carlier, 2011].

With the identification $\epsilon = \lambda^{-1}$, problem (7.8) is in fact equivalent to the R-D problem (7.4): compared to $\mathcal{L}_{BA}$ (7.5), the extra constraint on the second marginal of π in $\mathcal{L}_{EOT}$ (7.7) is redundant at the optimal ν. More precisely, Lemma D.1.1 shows that (we omit the notational dependence on μ when it is fixed):

$$\inf_{\nu \in \mathcal{P}(\mathcal{Y})} \mathcal{L}_{EOT}(\nu) = \inf_{\nu \in \mathcal{P}(\mathcal{Y})} \lambda^{-1} \mathcal{L}_{BA}(\nu) \qquad \text{and} \qquad \arg\min_{\nu \in \mathcal{P}(\mathcal{Y})} \mathcal{L}_{EOT}(\nu) = \arg\min_{\nu \in \mathcal{P}(\mathcal{Y})} \mathcal{L}_{BA}(\nu). \tag{7.9}$$

Existence of a minimizer holds under mild conditions, for instance if $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$ and $\rho(x, y)$ is a coercive lower semicontinuous function of $y - x$ [Csiszár, 1974a, p. 66].

7.2.3 Connection to Maximum-Likelihood Deconvolution

The connection between R-D and maximum-likelihood estimation has been observed in the information theory, machine learning and compression literature [Harrison and Kontoyiannis, 2008, Alemi et al., 2018, Ballé et al., 2017, Theis et al., 2017b, Yang et al., 2020a, Yang and Mandt, 2022]. Here, we bring attention to a basic equivalence between the R-D problem and maximum-likelihood deconvolution, where the connection is particularly natural under a quadratic distortion function. Also see [Rigollet and Weed, 2018] for a related discussion that inspired ours and extension to a non-quadratic distortion. We provide further insight from the view of variational learning and inference in Section D.4.

Maximum-likelihood deconvolution is a classical problem of non-parametric statistics and mixture models [Carroll and Hall, 1988, Lindsay and Roeder, 1993]. The deconvolution problem is concerned with estimating an unknown distribution α from noise-corrupted observations $X_1, X_2, ...$, where for each $i \in \mathbb{N}$, we have $X_i = Y_i + N_i$, $Y_i \overset{i.i.d.}{\sim} \alpha$, and N_i are i.i.d. independent noise variables with a known distribution. For concreteness, suppose all variables are $\mathbb{R}^d$ valued and the noise distribution is $\mathcal{N}(0, \sigma^2 \mathrm{I}_d)$ with Lebesgue density ϕ_{σ^2}. Denote the distribution of the observations X_i by μ. Then μ has a Lebesgue density given by the convolution $\alpha * \phi_{\sigma^2}(x) := \int \phi_{\sigma^2}(x-y)\alpha(dy)$. Here, we consider the population-level (instead of the usual sample-based) maximum-likelihood estimator (MLE) for α:

$$\nu^* = \underset{\nu \in \mathcal{P}(\mathbb{R}^d)}{\arg\max} \int \log\left(\nu * \phi_{\sigma^2}(x)\right) \mu(dx), \tag{7.10}$$

and observe that $\nu^* = \alpha$. Plugging in the density $\phi_{\sigma^2}(x) \propto e^{-\frac{1}{2\sigma^2}\|x\|^2}$, we see that the MLE problem (7.10) is equivalent to the R-D problem (7.4) with $\rho(x,y) = \frac{1}{2}\|x-y\|^2, \lambda = \frac{1}{\sigma^2}$, and $\mathcal{L}_{BA}$ given by (7.6) in the form of a marginal log-likelihood. Thus the R-D problem has the interpretation of estimating a distribution from its noisy observations given through μ, assuming a Gaussian noise with variance $\frac{1}{\lambda}$.

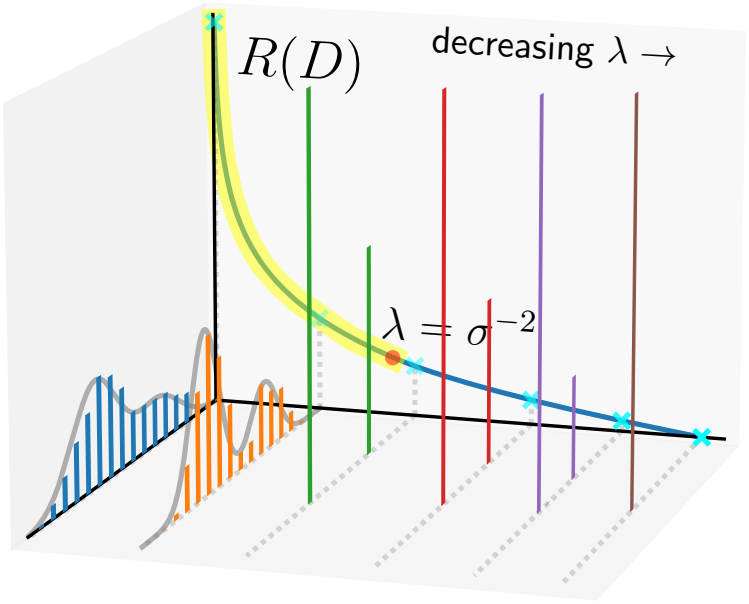


Figure 7.1: The $R(D)$ of a Gaussian mixture source, and the estimated optimal reproduction distributions ν^* (in bar plots) at varying R-D trade-offs. For any $\lambda \in [\sigma^{-2}, \infty)$, the corresponding $R(D)$ (yellow segment) is known analytically as is the optimal reproduction distribution ν^* (whose density is plotted in gray). For $\lambda \in (0, \sigma^{-2}]$, ν^* becomes singular and concentrated on two points, collapsing to the source mean as $\lambda \to 0$.

This connection suggests analytical solutions to the R-D problem for a variety of sources that arise from convolving an underlying distribution with Gaussian noise. Consider an R-D problem (7.4) with $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d, \rho(x, y) = \frac{1}{2}\|x - y\|^2$, and let the source μ be the convolution between an arbitrary measure $\alpha \in \mathcal{P}(\mathcal{Y})$ and Gaussian noise with known variance σ^2. E.g., using a discrete measure for α results in a Gaussian mixture source with equal covariance among its components. When $\lambda = \frac{1}{\sigma^2}$, we recover exactly the population-MLE problem (7.10) discussed earlier, which has the solution $\nu^* = \alpha$. While this allows us to obtain one point of $R(D)$, we can in fact extend this idea to any $\lambda \geq \frac{1}{\sigma^2}$ and obtain the analytical form for the corresponding *segment* of the R-D curve. Specifically, for any $\lambda \geq \frac{1}{\sigma^2}$, applying the summation rule for independent Gaussians reveals the source distribution μ as

$$\mu = \alpha * \mathcal{N}(0, \sigma^2) = \alpha * \mathcal{N}(0, \sigma^2 - \frac{1}{\lambda}) * \mathcal{N}(0, \frac{1}{\lambda}) = \alpha_\lambda * \mathcal{N}(0, \frac{1}{\lambda}), \quad \alpha_\lambda := \alpha * \mathcal{N}(0, \sigma^2 - \frac{1}{\lambda}),$$

i.e., as the convolution between another underlying distribution α_λ and independent noise with variance $\frac{1}{\lambda}$. A solution to the R-D problem (7.3) is then analogously given by $\nu^* = \alpha_\lambda$, with the corresponding optimal coupling given by $\nu^* \otimes \tilde{K}$, $\tilde{K}(y, dx) = \mathcal{N}(y, \frac{1}{\lambda})$. [1] Evaluating the distortion and mutual information of the coupling then yields the $R(D)$ point associated with λ. As an interesting side remark, note that the collection of optimal reproduction distributions $(\alpha_\lambda)_{\lambda \in [\sigma^{-2}, \infty)}$ corresponds to exactly the same probability path traced by variance-exploding diffusion models (VE-SDEs) [Song and Ermon, 2019, Song et al., 2020]. Fig. 7.1 illustrates the $R(D)$ of a toy Gaussian mixture source, along with the ν^* estimated by our proposed WGD algorithm (Sec. 7.4); note that ν^* transitions from continuous (a Gaussian mixture with smaller component variances) to singular (a mixture of two Diracs) at $\lambda = \sigma^{-2}$. See caption for more details.

7.3 Related Work

7.3.1 Blahut–Arimoto

The Blahut–Arimoto (BA) algorithm [Blahut, 1972, Arimoto, 1972] is the default method for computing $R(D)$ for a known and discrete case. For a fixed λ, BA carries out the optimization problem (7.3) via coordinate ascent. Starting from an initial measure $\nu^{(0)} \in \mathcal{P}(\mathcal{Y})$, the BA algorithm at step t computes an updated pair $(\nu^{(t+1)}, K^{(t+1)})$ as follows

$$\frac{dK^{(t+1)}(x, \cdot)}{d\nu^{(t)}}(y) = \frac{e^{-\lambda \rho(x,y)}}{\int e^{-\lambda \rho(x,y')} \nu^{(t)}(dy')}, \quad \forall x \in \mathcal{X}, \tag{7.11}$$

$$\nu^{(t+1)} = (\mu \otimes K^{(t+1)})_2. \tag{7.12}$$

[1] Here $\tilde{K}$ maps from the reproduction to the source alphabet, opposite to the kernel K elsewhere in the text.

When the alphabets are finite, the above computation can be carried out in matrix and vector operations, and the resulting sequence $\{(\nu^{(t)}, K^{(t)})\}_{t=1}^{\infty}$ can be shown to converge to an optimum of (7.3); cf. [Csiszár, 1974b, Csiszár, 1984]. When the alphabets are not finite, e.g., $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$, the BA algorithm no longer applies, as it is unclear how to digitally represent the measure ν and kernel K and to tractably perform the integrals required by the algorithm. The common workaround is to perform a discretization step and then apply BA on the resulting discrete problem.

One standard discretization method is to tile up the alphabets with small bins [Gray and Neuhoff, 1998]. This quickly becomes infeasible as the number of dimensions increases. We therefore consider discretizing the data space $\mathcal{X}$ to be the support of training data distribution μ^m, i.e., the discretized alphabet is the set of training samples; this can be justified by the consistency of the parametric R-D estimator $F^{\mathcal{H}}(\mu^m)$ [Harrison and Kontoyiannis, 2008]. It is less clear how to discretize the reproduction space $\mathcal{Y}$, especially in high dimensions. Since we work with $\mathcal{X} = \mathcal{Y}$, we will disretize $\mathcal{Y}$ similarly and use an n-element random subset of the training samples, as also considered by Lei et al. [2023a]. As we will show, this rather arbitrary placement of the support of ν results in poor performance, and can be significantly improved from our perspective of evolving particles.

7.3.2 Neural Network-Based Methods for Estimating $R(D)$

RD-VAE ([Yang and Mandt, 2022]): To overcome the limitations of the BA algorithm, Yang and Mandt [2022] proposed to parameterize the transition kernel K and reproduction distribution ν of the BA algorithm by neural density estimators [Papamakarios et al., 2021], and optimize the same objective (7.3) by (stochastic) gradient descent. They estimate (7.3) by Monte Carlo using joint samples $(X_i, Y_i) \sim \mu \otimes K$; in particular, the relative entropy can be written as $H(\mu \otimes K | \mu \otimes \nu) = \int \int \log \left(\frac{dK(x,\cdot)}{d\nu}(y) \right) K(x, dy)\mu(dx)$, where the integrand is

computed exactly via a density ratio. In practice, an alternative parameteriation is often used where the neural density estimators are defined on a lower dimensional latent space than the reproduction alphabet, and the resulting approach is closely related to VAEs [Kingma and Welling, 2013]. Yang and Mandt [2022] additionally propose a neural estimator for a lower bound on $R(D)$, based on a dual representation due to Csiszár [1974a]. **NERD [Lei et al., 2023a]:** Instead of working with the transition kernel K as in the RD-VAE, Lei et al. [2023a] considered optimizing the form of the rate functional in (7.6), via gradient descent on the parameters of ν parameterized by a neural network. Let $\nu^{\mathcal{Z}}$ be a base distribution over $\mathcal{Z} = \mathbb{R}^K$, such as the standard Gaussian, and $\omega : \mathcal{Z} \to \mathcal{Y}$ be a decoder network. The variational measure ν is then modeled as the image measure of $\nu^{\mathcal{Z}}$ under ω. To evaluate and optimize the objective (7.6), the intractable inner integral w.r.t. ν is replaced with a plug-in estimator, so that for a given $x \in \mathcal{X}$,

$$-\log\left(\int_{\mathcal{Y}} e^{-\lambda\rho(x,y)}\nu(dy)\right) \approx -\log\left(\frac{1}{n}\sum_{j=1}^{n} e^{-\lambda\rho(x,Y_j)}\right), \quad Y_j \sim \nu, j = 1, 2, ..., n. \tag{7.13}$$

After training, we estimate an R-D upper bound using n samples from ν (to be discussed in Sec. 7.4.4).

7.3.3 Other Related Work

Within information theory: Recent work by Wu et al. [2022] and Lei et al. [2023b] also note the connection between the R-D function and entropic optimal transport. Wu et al. [2022] compute the R-D function in the finite and known alphabet setting by solving a version of the EOT problem (7.8), whereas Lei et al. [2023b] numerically verify the equivalence (7.9) on a discrete problem and discuss the connection to scalar quantization. We also experimented with estimating $R(D)$ by solving the EOT problem (7.8), but found it computationally much more efficient to work with the rate functional (7.6), and we see the primary benefit of the

EOT connection as bringing in tools from statistical OT [Genevay et al., 2019, Mena and Niles-Weed, 2019, Rigollet and Stromme, 2022] for R-D estimation. **Outside of information theory:** Rigollet and Weed [2018] note a connection between the EOT projection problem (7.8) and maximum-likelihood deconvolution (7.10); our work complements their perspective by re-interpreting both problems through the equivalent R-D problem. Unbeknownst to us at the time, Yan et al. [2023] proposed similar algorithms to ours in the context of Gaussian mixture estimation, which we recognize as R-D estimation under quadratic distortion (see Sec. 7.2.3). Their work is based on gradient flow in the Fisher-Rao-Wasserstein (FRW) geometry [Chizat et al., 2018], which our hybrid algorithm can be seen as implementing. Yan et al. [2023] prove that, in an idealized setting with infinite particles, FRW gradient descent does not get stuck at local minima; by contrast, our convergence and sample-complexity results (Prop. 7.4.2, 7.4.3) hold for any finite number of particles. We additionally consider larger-scale problems and the stochastic optimization setting.

7.4 Proposed Method

For our algorithm, we require $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$ and ρ be continuously differentiable. We now introduce the gradient descent algorithm in Wasserstein space to solve the problems (7.4) and (7.8). We defer all proofs to Appendix D. To minimize a functional $\mathcal{L} : \mathcal{P}(\mathcal{Y}) \to \mathbb{R}$ over the space of probability measures, our algorithm essentially simulates the gradient flow [Ambrosio et al., 2008] of $\mathcal{L}$ and follows the trajectory of steepest descent in the Wasserstein geometry. In practice, we represent a measure $\nu^{(t)} \in \mathcal{P}(\mathcal{Y})$ by a collection of particles and at each time step update $\nu^{(t)}$ in a direction of steepest descent of $\mathcal{L}$ as given by its (negative) *Wasserstein gradient*. Denote by $\mathcal{P}_n(\mathbb{R}^d)$ the set of probability measures on $\mathbb{R}^d$ that are supported on at most n points. Our algorithm implements the parametric R-D estimator with the choice $\mathcal{H} = \mathcal{P}_n(\mathbb{R}^d)$ (see discussions at the end of Sec. 7.2.1).

7.4.1 Wasserstein Gradient Descent (WGD)

Abstractly, Wasserstein gradient descent updates the variational measure ν to its pushforward $\tilde{\nu}$ under the map $(\mathrm{id} - \gamma\Psi)$, for a function $\Psi : \mathbb{R}^d \to \mathbb{R}^d$ called the Wasserstein gradient of $\mathcal{L}$ at ν (see below) and a step size γ. To implement this scheme, we represent ν as a convex combination of Dirac measures, $\nu = \sum_{i=1}^n w_i \delta_{x_i}$ with locations $\{x_i\}_{i=1}^n \subset \mathbb{R}^d$ and weights $\{w_i\}_{i=1}^n$. The algorithm moves each particle x_i in the direction of $-\Psi(x_i)$, more precisely, $\tilde{\nu} = \sum_{i=1}^n w_i \delta_{x_i - \gamma\Psi(x_i)}$.

Algorithm 1 Wasserstein gradient descent

Inputs: Loss function $\mathcal{L} \in \{\mathcal{L}_{BA}, \mathcal{L}_{EOT}\}$; data distribution $\mu \in \mathcal{P}(\mathbb{R}^d)$; the number of particles $n \in \mathbb{N}$; total number of iterations $N \in \mathbb{N}$; step sizes $\gamma_1, \ldots, \gamma_N$; batch size $m \in \mathbb{N}$.
for $t = 1, \ldots, N$ **do**
  Pick an initial measure $\nu^{(0)} \in \mathcal{P}_n(\mathbb{R}^d)$, e.g., setting the particles to n random samples from μ.
  if support of μ contains more than m points **then**
    $\mu^m \leftarrow \frac{1}{m}\sum_{i=1}^m \delta_{x_i}$ for $x_1, \ldots, x_m$ independent samples from μ
    $\Psi^{(t)} \leftarrow$ Wasserstein gradient of $\mathcal{L}(\mu^m, \cdot)$ at $\nu^{(t-1)}$ {see Definition 7.4.1}
  else
    $\Psi^{(t)} \leftarrow$ Wasserstein gradient of $\mathcal{L}(\mu, \cdot)$ at $\nu^{(t-1)}$ {see Definition 7.4.1}
  end if
  $\nu^{(t)} \leftarrow \left(\mathrm{id} - \gamma_t \Psi^{(t)}\right)_{\#} \nu^{(t-1)}$ {"#" denotes pushforward}
end for
Return: $\nu^{(N)}$

Since the optimization objectives (7.4) and (7.8) appear as integrals w.r.t. the data distribution μ, we can also apply stochastic optimization and perform stochastic gradient descent on mini-batches with size m. This allows us to handle a very large or infinite amount of data samples, or when the source is continuous. We formalize the procedure in Algorithm 1.

The following gives a constructive definition of a Wasserstein gradient which forms the computational basis of our algorithm. In the literature, the Wasserstein gradient is instead usually defined as a Fréchet differential (cf. [Ambrosio et al., 2008, Definition 10.1.1]), but we emphasize that in smooth settings, the given definition recovers the one from the literature (cf. [Chizat, 2022, Lemma A.2]).

Definition 7.4.1. *For a functional $\mathcal{L} : \mathcal{P}(\mathcal{Y}) \to \mathbb{R}$ and $\nu \in \mathcal{P}(\mathcal{Y})$, we say that $V_{\mathcal{L}}(\nu) : \mathbb{R}^d \to \mathbb{R}$ is a first variation of $\mathcal{L}$ at ν if*

$$\lim_{\varepsilon \to 0} \frac{\mathcal{L}((1-\varepsilon)\nu + \varepsilon\tilde{\nu}) - \mathcal{L}(\nu)}{\varepsilon} = \int V_{\mathcal{L}}(\nu)\, d(\tilde{\nu} - \nu) \quad \text{for all } \tilde{\nu} \in \mathcal{P}(\mathcal{Y}).$$

We call its (Euclidean) gradient $\nabla V_{\mathcal{L}}(\nu) : \mathbb{R}^d \to \mathbb{R}^d$, if it exists, the Wasserstein gradient of $\mathcal{L}$ at ν.

For $\mathcal{L} = \mathcal{L}_{EOT}$, the first variation is given by the Kantorovich potential, which is the solution of the convex dual of $\mathcal{L}_{EOT}$ and commonly computed by Sinkhorn's algorithm [Peyré and Cuturi, 2019, Nutz, 2021]. Specifically, let (φ^ν, ψ^ν) be potentials for $\mathcal{L}_{EOT}(\mu, \nu)$. Then $V_{\mathcal{L}}(\nu) = \psi^\nu$ is the first variation w.r.t. ν (cf. [Carlier et al., 2022, equation (20)]), and hence $\nabla\psi^\nu$ is the Wasserstein gradient. This gradient exists whenever ρ is differentiable and the marginals are sufficiently light-tailed; we give details in Sec. D.3.1 of Appendix D. For $\mathcal{L} = \mathcal{L}_{BA}$, the first variation can be computed explicitly. As derived in Sec. D.3.1 of Appendix D, the first variation at ν is

$$\psi^\nu(y) = \int -\frac{\exp(-\lambda\rho(x,y))}{\int \exp(-\lambda\rho(x,\tilde{y}))\nu(d\tilde{y})}\mu(dx)$$

and then the Wasserstein gradient is $\nabla\mathcal{L}_{BA}(\nu) = \nabla\psi^\nu$. We observe that $\psi^\nu(y)$ is computationally cheap; it corresponds to running a single iteration of Sinkhorn's algorithm. By contract, finding the potential for $\mathcal{L}_{EOT}$ requires running Sinkhorn's algorithm to convergence.

Like the usual Euclidean gradient, the Wasserstein gradient can be shown to possess a linearization property, whereby the loss functional is reduced by taking a small enough step along its Wasserstein gradient. Following [Carlier et al., 2022], we state it as follows: for any

$\tilde{\nu} \in \mathcal{P}(\mathcal{Y})$ and $\pi \in \Pi(\nu, \tilde{\nu})$,

$$\begin{aligned} \mathcal{L}(\tilde{\nu}) - \mathcal{L}(\nu) = \int (y-x)^\top \nabla V_{\mathcal{L}}(\nu)(x)\, \pi(dx, dy) + o\left(\int \|y-x\|^2 \, \pi(dx, dy)\right), \\ \left| \int \|\nabla V_{\mathcal{L}}(\nu)\|^2 \, d\nu - \int \|\nabla V_{\mathcal{L}}(\tilde{\nu})\|^2 \, d\tilde{\nu} \right| \leq C W_2(\nu, \tilde{\nu}). \end{aligned} \tag{7.14}$$

The first line of (7.14) is proved for $\mathcal{L}_{EOT}$ in [Carlier et al., 2022, Proposition 4.2] in the case that the marginals are compactly supported and ρ is twice continuously differentiable. In this setting, the second line of (7.14) follows using $a^2 - b^2 = (a+b)(a-b)$ and a combination of boundedness and Lipschitz continuity of $\nabla V_{\mathcal{L}}$, see [Carlier et al., 2022, Proposition 2.2 and Corollary 2.4].

The linearization property given by (7.14) enables us to show that Wasserstein gradient descent for $\mathcal{L}_{EOT}$ and $\mathcal{L}_{BA}$ converges to a stationary point under mild conditions:

Proposition 7.4.2 (Convergence of Wasserstein gradient descent)**.** *Let $\gamma_1 \geq \gamma_2 \geq \cdots \geq 0$ satisfy $\sum_{k=1}^{\infty} \gamma_k = \infty$ and $\sum_{k=1}^{\infty} \gamma_k^2 < \infty$. Let $\mathcal{L} : \mathcal{P}(\mathbb{R}^d) \to \mathbb{R}$ be Wasserstein differentiable in the sense that (7.14) holds. Denoting by $\nu^{(t)}$ the steps in Algorithm 1, and suppose that $\mathcal{L}(\nu^{(0)})$ is finite and $\int \|\nabla V_{\mathcal{L}}(\nu^{(t)})\|^2 \, d\nu^{(t)}$ is bounded. Then*

$$\lim_{t \to \infty} \int \|\nabla V_{\mathcal{L}}(\nu^{(t)})\|^2 \, d\nu^{(t)} = 0.$$

7.4.2 Hybrid Algorithm

A main limitation of the BA algorithm is that the support of $\nu^{(t)}$ is restricted to that of the (possibly bad) initialization $\nu^{(0)}$. On the other hand, Wasserstein gradient descent (Algorithm 1) only evolves the particle locations of $\nu^{(t)}$, but not the weights, which are fixed to be uniform by default. We therefore consider a hybrid algorithm where we alternate between WGD and the BA update steps, allowing us to optimize the particle weights as

well. Experimentally, this translates to faster convergence than the base WGD algorithm (Sec. 7.5.1). Note however, unlike WGD, the hybrid algorithm does not directly lend itself to the stochastic optimization setting, as BA updates on mini-batches no longer guarantee monotonic improvement in the objective and can lead to divergence. We treat the convergence of the hybrid algorithm in Sec. D.3.4 of Appendix D.

7.4.3 Sample Complexity

Let $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$ and $\rho(x, y) = \|x - y\|^2$. Leveraging work on the statistical complexity of EOT [Mena and Niles-Weed, 2019], we obtain finite-sample bounds for the theoretical estimators implemented by WGD in terms of the number of particles and source samples. The bounds hold for both the R-D problem (7.4) and EOT projection problem (7.8) as they share the same optimizers (see Sec. 7.2.2), and strengthen existing asymptotic results for empirical R-D estimators [Harrison and Kontoyiannis, 2008]. We note that a recent result by Rigollet and Stromme [2022] might be useful for deriving alternative bounds under distortion functions other than the quadratic.

Proposition 7.4.3. *Let μ be σ^2-subgaussian. Then every optimizer ν^* of (7.4) and (7.8) is also σ^2-subgaussian. Consider $\mathcal{L} := \mathcal{L}_{EOT}$. For a constant C_d only depending on d, we have*

$$\begin{aligned}
\left| \min_{\nu \in \mathcal{P}(\mathbb{R}^d)} \mathcal{L}(\mu, \nu) - \min_{\nu_n \in \mathcal{P}_n(\mathbb{R}^d)} \mathcal{L}(\mu, \nu_n) \right| &\leq C_d\, \epsilon \left(1 + \frac{\sigma^{\lceil 5d/2 \rceil + 6}}{\epsilon^{\lceil 5d/4 \rceil + 3}}\right) \frac{1}{\sqrt{n}}, \\
\mathbb{E}\left[\left| \min_{\nu \in \mathcal{P}(\mathbb{R}^d)} \mathcal{L}(\mu, \nu) - \min_{\nu \in \mathcal{P}(\mathbb{R}^d)} \mathcal{L}(\mu^m, \nu) \right| \right] &\leq C_d\, \epsilon \left(1 + \frac{\sigma^{\lceil 5d/2 \rceil + 6}}{\epsilon^{\lceil 5d/4 \rceil + 3}}\right) \frac{1}{\sqrt{m}}, \\
\mathbb{E}\left[\left| \min_{\nu \in \mathcal{P}(\mathbb{R}^d)} \mathcal{L}(\mu, \nu) - \min_{\nu_n \in \mathcal{P}_n(\mathbb{R}^d)} \mathcal{L}(\mu^m, \nu_n) \right| \right] &\leq C_d\, \epsilon \left(1 + \frac{\sigma^{\lceil 5d/2 \rceil + 6}}{\epsilon^{\lceil 5d/4 \rceil + 3}}\right) \left(\frac{1}{\sqrt{m}} + \frac{1}{\sqrt{n}} \right),
\end{aligned}$$

for all $n, m \in \mathbb{N}$, where $\mathcal{P}_n(\mathbb{R}^d)$ is the set of probability measures over $\mathbb{R}^d$ supported on at most n points, μ^m is the empirical measure of μ with m independent samples and the expectation $\mathbb{E}[\cdot]$ is over these samples. The same inequalities hold for $\mathcal{L} := \lambda^{-1}\mathcal{L}_{BA}$, with the identification

$\epsilon = \lambda^{-1}$.

7.4.4 Estimation of Rate and Distortion

Here, we describe our estimator for an upper bound $(\mathcal{D}, \mathcal{R})$ of $R(D)$ after solving the unconstrained problem (7.3). We provide more details in Sec. D.2 of the Supplementary Material.

For any given pair of ν and K, we always have that $\mathcal{D} := \int \rho d(\mu \otimes K)$ and $\mathcal{R} := H(\mu \otimes K | \mu \otimes \nu)$ lie on an upper bound of $R(D)$ [Berger, 1971]. The two quantities can be estimated by standard Monte Carlo provided we can sample from $\mu \otimes K$ and evaluate the density $\frac{d\mu \otimes K}{d\mu \otimes \nu}(x, y) = \frac{dK(x,\cdot)}{d\nu}(y)$.

When only ν is given, e.g., obtained from optimizing (7.6) with WGD or NERD, we estimate an R-D upper bound as follows. As in the BA algorithm, we construct a kernel K_ν similarly to (7.11), i.e., $\frac{dK_\nu(x,\cdot)}{d\nu}(y) = \frac{e^{-\lambda\rho(x,y)}}{\int e^{-\lambda\rho(x,\tilde{y})}\nu(d\tilde{y})}$; then we estimate $(\mathcal{D}, \mathcal{R})$ using the pair (ν, K_ν) as described earlier.

As NERD uses a continuous ν, we follow [Lei et al., 2023a] and approximate it with its n-sample empirical measure to estimate $(\mathcal{D}, \mathcal{R})$. A limitation of NERD, BA, and our method is that they tend to converge to a rate estimate of at most $\log(n)$, where n is the support size of ν. This is because as the algorithms approach an n-point minimizer ν_n^* of the R-D problem, the rate estimate $\mathcal{R}$ approaches the mutual information of $\mu \otimes K_{\nu_n^*}$, which is upper-bounded by $\log(n)$ [Eckstein and Nutz, 2022]. In practice, this means if a target point of $R(D)$ has rate r, then we need $n \geq e^r$ to estimate it accurately.

7.4.5 Computational Considerations

Common to all the aforementioned methods is the evaluation of a pairwise distortion matrix between m points in $\mathcal{X}$ and n points in $\mathcal{Y}$, which usually has a cost of $\mathcal{O}(mnd)$ for a d-dimensional source. While RD-VAE uses $n = 1$ (in the reparameterization trick), the other methods (BA, WGD, NERD) typically use a much larger n and thus has the distortion computation as their main computation bottleneck. Compared to BA and WGD, the neural methods (RD-VAE, NERD) incur additional computation from neural network operations, which can be significant for large networks.

For NERD and WGD (and BA), the rate estimate upper bound of $\log(n)$ nats/sample (see Sec. 7.4.4) can present computational challenges. To target a high-rate setting, a large number of ν particles (high n) is required, and care needs to be taken to avoid running out of memory during the distortion matrix computation (one possibility is to use a small batch size m with stochastic optimization).

7.5 Experiments

We compare the empirical performance of our proposed method (WGD) and its hybrid variant with Blahut–Arimoto (BA) [Blahut, 1972, Arimoto, 1972], RD-VAE [Yang and Mandt, 2022], and NERD [Lei et al., 2022] on the tasks of maximum-likelihood deconvolution and estimation of R-D upper bounds. While we experimented with WGD for both $\mathcal{L}_{BA}$ and $\mathcal{L}_{EOT}$, we found the former to be 10 to 100 times faster computationally while giving similar or better results; we therefore focus on WGD for $\mathcal{L}_{BA}$ in discussions below. For the neural-network baselines, we use the same (or as similar as possible) network architectures as in the original work [Yang and Mandt, 2022, Lei et al., 2023a]. We use the Adam optimizer for all gradient-based methods, except we use simple gradient descent with a decaying step size in Sec. 7.5.1 to

better compare the convergence speed of WGD and its hybrid variant. Further experiment details and results are given in Sec. D.5 of Appendix D.

7.5.1 Deconvolution

To better understand the behavior of the various algorithms, we apply them to a deconvolution problem with known ground truth (see Sec. 7.2.3). We adopt the Gaussian noise as before, letting α be the uniform measure on the unit circle in $\mathbb{R}^2$ and the source $\mu = \alpha * \mathcal{N}(0, \sigma^2)$ with $\sigma^2 = 0.1$.

We use $n = 20$ particles for BA, NERD, WGD and its hybrid variant. We use a two-layer network for NERD and RD-VAE with some hand-tuning (we replace the softplus activation in the original RD-VAE network by ReLU as it led to difficulty in optimization). Fig. 7.2 plots the resulting loss curves and shows that the proposed algorithms converge the fastest to the ground truth value $OPT := \mathcal{L}(\alpha)$. In Fig. 7.3, we visualize the final $\nu^{(t)}$ at the end of training, compared to the ground truth $\nu^* = \alpha$ supported on the circle (colored in cyan). Note that we initialize $\nu^{(0)}$ for BA, WGD, and its hybrid variant to the same n random data samples. While BA is stuck with the randomly initialized particles and assigns large weights to those closer to the circle, WGD learns to move the particles to uniformly cover the circle. The hybrid algorithm, being able to reweight particles to reduce their transportation cost, learns a different solution where a cluster of particles covers the top-left portion of the circle with small weights while the remaining particles evenly covers the rest. Unlike our particle-based methods, the neural methods generally struggle to place the support of their ν exactly on the circle.

We additionally compare how the performance of BA, NERD, and the proposed algorithms scale to higher dimensions and a higher λ (corresponding to lower entropic regularization in $\mathcal{L}_{EOT}$). Fig. 7.4 plots the gap between the converged and the optimal losses for the

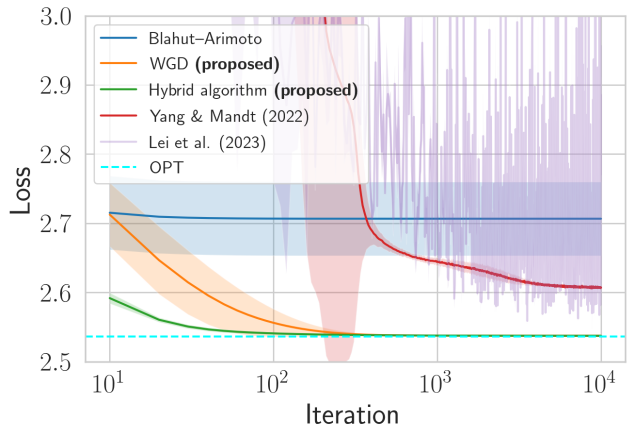


Figure 7.2: Losses over iterations. Shading corresponds to one standard deviation over random initializations.

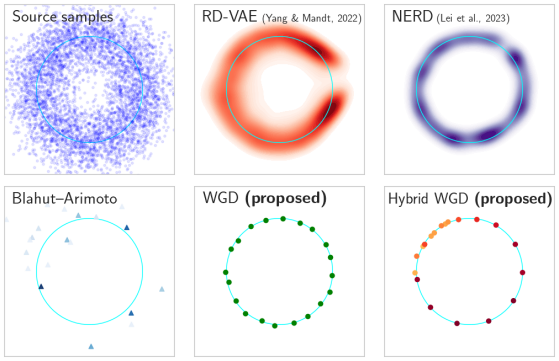


Figure 7.3: Visualizing μ samples (top left), as well as the ν returned by various algorithms compared to the ground truth ν^* (cyan).

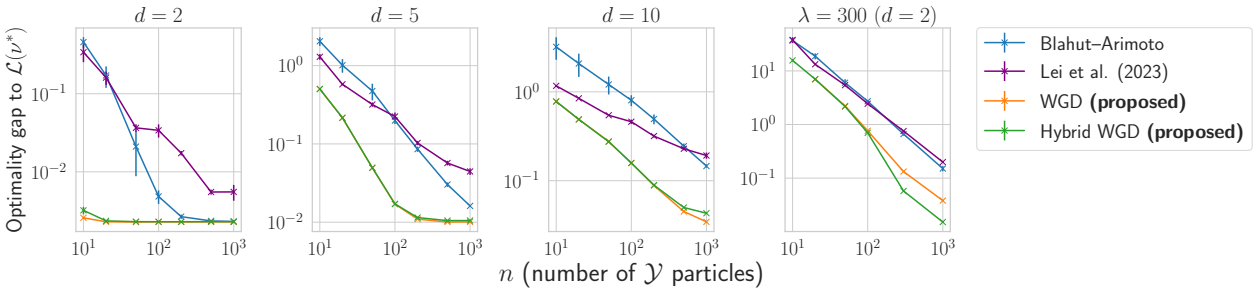


Figure 7.4: Optimality gap v.s. the number of particles n used, on deconvolution problems with different dimension d and distortion multiplier λ. The first three panels fix $\lambda = \sigma^{-2}$ and increase d, and the right-most panel corresponds to the 2-D problem with a higher λ (denoising with a narrower Gaussian kernel). Overall the WGD methods attain higher accuracy for a given budget of n.

algorithms, and demonstrates the proposed algorithms to be more particle-efficient and scale more favorably than the alternatives which also use n particles in the reproduction space. We additionally visualize how the converged particles for our methods vary across the R-D trade-off in Fig. D.1 of the Appendix.

7.5.2 Higher-Dimensional Data

We perform $R(D)$ estimation on higher-dimensional data, including the *physics* and *speech* datasets from [Yang and Mandt, 2022] and MNIST [LeCun et al., 1998]. As the memory cost

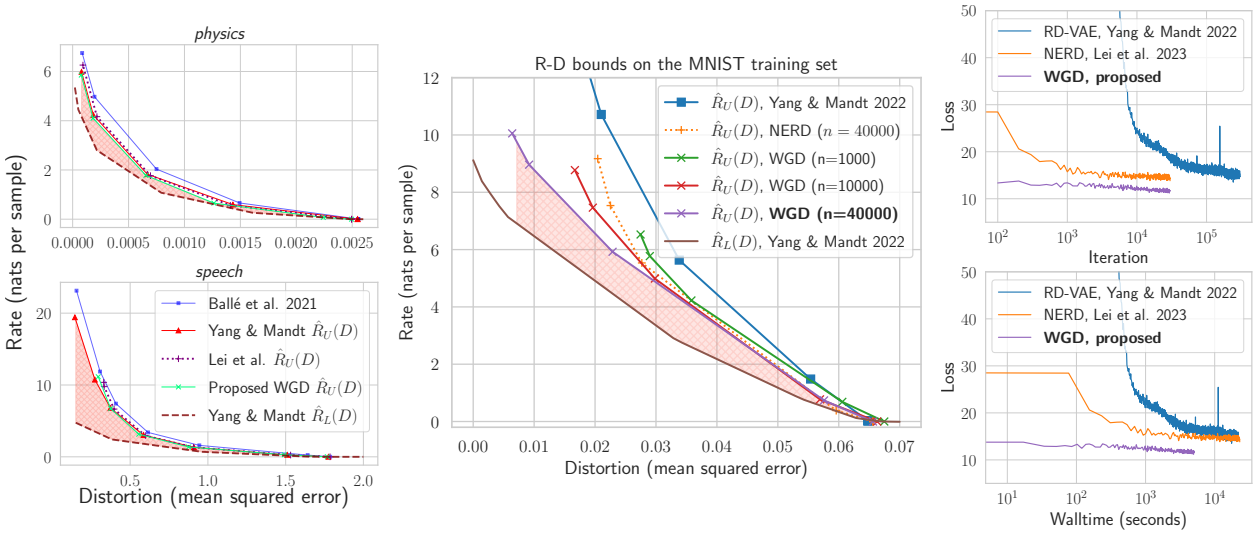


Figure 7.5: **Left, Middle:** R-D bound estimates on the physics, speech datasets [Yang and Mandt, 2022] and MNIST training set. **Right:** Example speed comparisons of WGD and neural upper bound methods on MNIST, with WGD converging at least an order of magnitude faster. On each of the dataset we also include an R-D *lower* bound estimated using the method of [Yang and Mandt, 2022].

to operating on the full datasets becomes prohibitive, we focus on NERD, RD-VAE, and WGD using mini-batch stochastic gradient descent. BA and hybrid WGD do not directly apply in the stochastic setting, as BA updates on random mini-batches can lead to divergence (as discussed in Sec. 7.4.2).

Fig. 7.5 plots the estimated R-D bounds on the datasets, and compares the convergence speed of WGD and neural methods in both iteration count and compute time. Overall, we find WGD to require minimal tuning and obtains the tightest R-D upper bounds within the $\log(n)$ rate limit (see Sec. 7.4.4), and consistently obtains tighter bounds than NERD given the same computation budget.

7.6 Discussions

In this work, we leverage tools from optimal transport to develop a new approach for estimating the rate-distortion function in the continuous setting. Compared to state-of-the-art neural approaches [Yang and Mandt, 2022, Lei et al., 2022], our Wasserstein gradient descent algorithm offers complementary strengths: 1) It requires a single main hyperparameter n (the number of particles) and no network architecture tuning; and 2) empirically we found it to converge significantly faster and rarely end up in bad local optima; increasing n almost always yielded an improvement (unless the bound is already close to being tight). From a modeling perspective, a particle representation may be inherently more efficient when the optimal reproduction distribution is singular or has many disconnected modes (shown, e.g., in Figs. 7.1 and 7.3). However, like NERD [Lei et al., 2022], our method has a fundamental limitation – it requires an n that is exponential in the rate of the targeted $R(D)$ point to estimate it accurately (see Sec. 7.4.4). Thus, a neural method like RD-VAE [Yang and Mandt, 2022] may still be preferable on high-rate sources, while our particle-based method stands as a state-of-the-art solution on lower-rate sources with substantially less tuning or computation requirements.

Besides R-D estimation, our algorithm also applies to the mathematically equivalent problems of maximum likelihood deconvolution and projection under an entropic optimal transport (EOT) cost, and may find other connections and applications. Indeed, the R-D problem is further related to the statistical problems of selecting reference priors [Lafferty and Wasserman, 2013] or minimally informative likelihoods [Yuan and Clarke, 1999]. The EOT projection view of our algorithm is also related to optimization-based approaches to sampling [Wibisono, 2018], variational inference [Liu and Wang, 2016], and distribution compression [Shetty et al., 2021]. Our particle-based algorithm generalizes optimal quantization (corresponding to $\epsilon = 0$ in $\mathcal{L}_{EOT}$, i.e., projecting the source under the Wasserstein distance [Graf and Luschgy, 2007, Gray, 2013]) to incorporate a rate constraint ($\epsilon > 0$), and it would be interesting to explore

the use of the resulting rate-distortion optimal quantizer for practical data compression and communication.

Chapter 8

Conclusion and Future Topics

The connection between density estimation and lossy compression under the rate-distortion criterion goes back to [Harrison and Kontoyiannis, 2008] (and perhaps earlier), where the latter problem is equivalent to maximizing the expected marginal log-likelihood of an appropriately defined latent-variable model. This connection conceptually enables the transfer of ideas between lossy compression and probabilistic modeling, mirroring the tight connection between lossless compression and probabilistic modeling [Kraft, 1949, McMillan, 1956]. However, Shannon-style block coding (which achieves the theoretically optimal R-D performance) is computationally infeasible and rarely used, and additional effort is often required to translate the ideas and advances in variational inference/learning to practical gains in lossy compression performance. The proposal in Chapter 4 can be seen as one such effort, where the variational inference ("encoding") distribution in SGA is designed to work with the quantized latent space of nonlinear transform coding. Relative entropy coding and channel simulation [Theis and Yosri, 2021, Flamich et al., 2020a], avoid quantization by performing stochastic encoding, and can be seen as another effort to fundamentally reduce the problem of lossy source coding to that of probabilistic machine learning [1]. Another advantage of stochastic encoding is the

[1] Also see [Yang et al., 2020b] with a shared goal but uses a different quantization scheme than in nonlinear transform coding.

potential for better realism [Blau and Michaeli, 2018, Blau and Michaeli, 2019, Theis and Agustsson, 2021], an important metric that is not discussed in this thesis but is receiving increasing attention [Mentzer et al., 2020, Yan et al., 2021, Qiu et al., 2024].

In addition to furthering the connection between lossy compression and machine learning, a few other potential research topics deserve mentioning:

- Iterative inference, e.g., using SGA [Yang et al., 2020a], significantly improves compression performance, but at a much higher encoding computation cost compared to amortized inference. Can we learn a procedure to speed up iterative inference, potentially based on ideas from meta-learning [Marino et al., 2018] or implicit differentiation [Chang et al., 2022] in an end-to-end manner?

- Implicit neural representations (INRs) [Sitzmann et al., 2020] have emerged as a new paradigm for signal representation and lossy compression, and many INR-based compression approaches [Dupont et al., 2021, 2022, Mehta et al., 2021] share conceptual similarities to nonlinear transform coding (NTC) [Ballé et al., 2021] combined with iterative encoding. Can we gain a deeper understanding of the INR-based approach to lossy compression from the perspective of NTC and rate-distortion theory? We offer some preliminary thoughts in [Pham et al., 2023], suggesting that the distinction between the INR-based approach ("model compression") and the NTC approach ("data compression") may not be so clear, and it is possible to get the best of both worlds.

- Diffusion models have shown impressive performance in generative modeling tasks. Given their natural connection to progressive compression [Ho et al., 2020, Shu and Ermon, 2022, Theis et al., 2022], they may lead to promising new compression schemes, one of which we explore in [Yang et al., 2025]. Similarly, diffusion models can be applied in the manner of Chapter 6 and [Theis et al., 2022] to yield new and improved upper bounds on the R-D function, with the ability to produce an entire R-D curve with a

single pre-trained model. Furthermore, there exist an interesting connection between the intrinsic data dimensionality and the learned diffusion model (in the low noise limit) [Batzolis et al., 2022], and a similar connection between the intrinsic data dimensionality and the R-D function (in the low distortion limit) [Kégl, 2002, Kawabata and Dembo, 1994], suggesting that a unified perspective may be possible.

Bibliography

D. Minnen, J. Ballé, and G. D. Toderici. Joint Autoregressive and Hierarchical Priors for Learned Image Compression. In *Advances in Neural Information Processing Systems 31*, pages 10771–10780. 2018.

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *International Conference on Learning Representations*, 2017.

Eastman Kodak. Kodak lossless true color image suite (PhotoCD PCD0992), 1993. URL http://r0k.us/graphics/kodak.

Fabrice Bellard. BPG specification, 2014. URL https://bellard.org/bpg/bpg_spec.txt. (accessed June 3, 2020).

J. Ballé, P. A. Chou, D. Minnen, S. Singh, N. Johnston, E. Agustsson, S. J. Hwang, and G. Toderici. Nonlinear transform coding. *IEEE Trans. on Special Topics in Signal Processing*, 15, 2021.

Yibo Yang and Stephan Mandt. Towards empirical sandwich bounds on the rate-distortion function. In *International Conference on Learning Representations*, 2022.

R. Blahut. Computation of channel capacity and rate-distortion functions. *IEEE Transactions on Information Theory*, 18(4):460–473, 1972. doi: 10.1109/TIT.1972.1054855.

Suguru Arimoto. An algorithm for computing the capacity of arbitrary discrete memoryless channels. *IEEE Transactions on Information Theory*, 18(1):14–20, 1972.

Leon Gordon Kraft. *A device for quantizing, grouping, and coding amplitude-modulated pulses*. PhD thesis, Massachusetts Institute of Technology, 1949.

Brockway McMillan. Two inequalities implied by unique decipherability. *IRE Transactions on Information Theory*, 2(4):115–116, 1956.

Christopher M Bishop. *Pattern Recognition and Machine Learning*. Information science and statistics. Springer, 2006. doi: 10.1117/1.2819119.

L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. In *International Conference on Learning Representations*, Apr 2016. URL http://arxiv.org/abs/1511.01844.

BusinessWire. Data creation and replication will grow at a faster rate than installed storage capacity. *URL: https://www.businesswire.com/news/home/20210324005175/en/Data-Creation-and-Replication-Will-Grow-at-a-Faster-Rate-Than-Installed-Storage-Capacity-According-to-the-IDC-Global-DataSphere-and-StorageSphere-Forecasts*, 2021.

Jorma Rissanen and Glen G Langdon. Arithmetic coding. *IBM Journal of research and development*, 23(2):149–162, 1979.

Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.

Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE transactions on Information Theory*, 24(5):530–536, 1978.

Peter D Grünwald. *The minimum description length principle*. MIT press, 2007.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, 2015.

C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423, 1948.

D. Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, 2014.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 22(1):2617–2680, 2021.

Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Practical full resolution learned lossless image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10629–10638, 2019.

James Townsend, Tom Bird, and David Barber. Practical lossless compression with latent variables using bits back coding. *arXiv preprint arXiv:1901.04866*, 2019a.

Emiel Hoogeboom, Jorn Peters, Rianne van den Berg, and Max Welling. Integer discrete flows and lossless compression. In *Advances in Neural Information Processing Systems*, pages 12134–12144, 2019.

Dustin Tran, Keyon Vafa, Kumar Agrawal, Laurent Dinh, and Ben Poole. Discrete flows: Invertible generative models of discrete data. In *Advances in Neural Information Processing Systems*, pages 14719–14728, 2019.

Jonathan Ho, Evan Lohn, and Pieter Abbeel. Compression with flows via local bits-back coding. In *Advances in Neural Information Processing Systems*, pages 3874–3883, 2019a.

Emiel Hoogeboom, Alexey A Gritsenko, Jasmijn Bastings, Ben Poole, Rianne van den Berg, and Tim Salimans. Autoregressive diffusion models. *arXiv preprint arXiv:2110.02037*, 2021a.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Karol Gregor, Frederic Besse, Danilo Jimenez Rezende, Ivo Danihelka, and Daan Wierstra. Towards conceptual compression. *Neural Info. Process. Sys.*, 29:3549–3557, 2016.

J. Ballé, V. Laparra, and E. P. Simoncelli. End-to-end optimization of nonlinear transform codes for perceptual quality. In *Picture Coding Symposium*, 2016.

L. Theis, W. Shi, A. Cunningham, and F. Huszár. Lossy Image Compression with Compressive Autoencoders. In *International Conference on Learning Representations*, 2017a.

E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte, and L. Van Gool. Generative adversarial networks for extreme learned image compression. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 221–231, 2019.

Fabian Mentzer, George D Toderici, Michael Tschannen, and Eirikur Agustsson. High-fidelity generative image compression. *Advances in Neural Information Processing Systems*, 33, 2020.

Lucas Theis, Tim Salimans, Matthew D Hoffman, and Fabian Mentzer. Lossy compression with gaussian diffusion. *arXiv preprint arXiv:2206.08889*, 2022.

Noor Fathima Khanum Mohamed Ghouse, Jens Petersen, Auke J. Wiggers, Tianlin Xu, and Guillaume Sautiere. Neural image compression with a diffusion-based decoder, 2023. URL https://openreview.net/forum?id=4Jq0XWCZQel.

Emiel Hoogeboom, Eirikur Agustsson, Fabian Mentzer, Luca Versari, George Toderici, and Lucas Theis. High-fidelity image compression with score-based generative models. *arXiv preprint arXiv:2305.18231*, 2023.

Ruihan Yang and Stephan Mandt. Lossy image compression with conditional diffusion models. *Advances in Neural Information Processing Systems*, 36, 2023a.

Yibo Yang, Justus Will, and Stephan Mandt. Progressive compression with universally quantized diffusion models. In *International Conference on Learning Representations*, 2025.

D. Minnen and S. Singh. Channel-wise autoregressive entropy models for learned image compression. In *IEEE International Conference on Image Processing (ICIP)*, 2020.

Yibo Yang, Robert Bamler, and Stephan Mandt. Improving inference for neural image compression. In *Neural Information Processing Systems (NeurIPS), 2020*, 2020a.

Dailan He, Ziming Yang, Weikun Peng, Rui Ma, Hongwei Qin, and Yan Wang. Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5718–5727, 2022.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2021.

Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *arXiv preprint arXiv:2210.13438*, 2022.

Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. Dvc: An end-to-end deep video compression framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11006–11015, 2019.

Eirikur Agustsson, David Minnen, Nick Johnston, Johannes Balle, Sung Jin Hwang, and George Toderici. Scale-space flow for end-to-end optimized video compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8503–8512, 2020.

Ruihan Yang, Yibo Yang, Joseph Marino, and Stephan Mandt. Hierarchical autoregressive modeling for neural video compression. In *International Conference on Learning Representations*, 2021.

Fabian Mentzer, George Toderici, David Minnen, Sung-Jin Hwang, Sergi Caelles, Mario Lucic, and Eirikur Agustsson. Vct: A video compression transformer. *arXiv preprint arXiv:2206.07307*, 2022.

André FR Guarda, Nuno MM Rodrigues, and Fernando Pereira. Point cloud coding: Adopting a deep learning-based approach. In *2019 Picture Coding Symposium (PCS)*, pages 1–5. IEEE, 2019.

Maurice Quach, Jiahao Pang, Dong Tian, Giuseppe Valenzise, and Frédéric Dufaux. Survey on deep learning-based point cloud compression. *Frontiers in Signal Processing*, 2022.

Danhang Tang, Saurabh Singh, Philip A Chou, Christian Hane, Mingsong Dou, Sean Fanello, Jonathan Taylor, Philip Davidson, Onur G Guleryuz, Yinda Zhang, et al. Deep implicit volume compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1293–1303, 2020.

Thomas Bird, Johannes Ballé, Saurabh Singh, and Philip A Chou. 3d scene compression through entropy penalized neural representation functions. In *2021 Picture Coding Symposium (PCS)*, pages 1–5. IEEE, 2021.

David Minnen. Current Frontiers In Neural Image Compression: The Rate-Distortion-Computation Trade-Off And Optimizing For Subjective Visual Quality, 2021. URL https://www.youtube.com/watch?v=84XkOlBhOas.

Debargha Mukherjee. Challenges in incorporating ML in a mainstream nextgen video codec. *CLIC Workshop and Challenge on Learned Image Compression*, 2022. URL https://storage.googleapis.com/clic2022_public/slides/Challenges%20in%20incorporating%20ML%20in%20a%20practical%20Nextgen%20Video%20Codec.pdf.

CE Shannon. Coding theorems for a discrete source with a fidelity criterion. *IRE Nat. Conv. Rec., March 1959*, 4:142–163, 1959.

Ioannis Kontoyiannis. Pointwise redundancy in lossy data compression and universal lossy data compression. *IEEE Transactions on Information Theory*, 46(1):136–152, 2000.

Victoria Kostina. When is shannon's lower bound tight at finite blocklength? In *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 982–989. IEEE, 2016.

Y. Blau and T. Michaeli. The perception-distortion tradeoff. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6228–6237, 2018.

Yochai Blau and Tomer Michaeli. Rethinking lossy compression: The rate-distortion-perception tradeoff. In *International Conference on Machine Learning*, pages 675–685. PMLR, 2019.

Nick Johnston, Elad Eban, Ariel Gordon, and Johannes Ballé. Computationally efficient neural image compression. *arXiv preprint arXiv:1912.08771*, 2019.

Yibo Yang and Stephan Mandt. Computationally-efficient neural image compression with shallow decoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 530–540, 2023b.

Yang Qiu, Aaron B Wagner, Johannes Ballé, and Lucas Theis. Wasserstein distortion: Unifying fidelity and realism. In *2024 58th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2024.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *Iclr*, 2(5):6, 2017.

Imre Csiszár. On an extremum problem of information theory. *Studia Scientiarum Mathematicarum Hungarica*, 9, 01 1974a.

Luigi Ambrosio, Nicola Gigli, and Giuseppe Savaré. *Gradient flows in metric spaces and in the space of probability measures*. Lectures in Mathematics ETH Zürich. Birkhäuser Verlag, Basel, second edition, 2008.

Matthew T. Harrison and Ioannis Kontoyiannis. Estimation of the rate–distortion function. *IEEE Transactions on Information Theory*, 54(8):3757–3762, Aug 2008. ISSN 0018-9448. doi: 10.1109/tit.2008.926387. URL http://dx.doi.org/10.1109/TIT.2008.926387.

Gonzalo Mena and Jonathan Niles-Weed. Statistical bounds for entropic optimal transport: sample complexity and the central limit theorem. *Advances in Neural Information Processing Systems*, 32, 2019.

Allen Gersho and Robert M Gray. *Vector quantization and signal compression*, volume 159. Springer Science & Business Media, 2012.

Matthew J Beal and Zoubin Ghahramani. The variational Bayesian EM algorithm for incomplete data: with application to scoring graphical model structures. In *Bayesian Statistics 7: Proceedings of the Seventh Valencia International Meeting*, volume 7, pages 453–464. Oxford University Press, University of London, University College London (United Kingdom), 2003a.

Yury Polyanskiy and Yihong Wu. Information theory: From coding to learning. *Book draft*, 2022.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*, volume 2. John Wiley & Sons, 2006.

Yibo Yang, Stephan Mandt, and Lucas Theis. An introduction to neural data compression. *Foundations and Trends® in Computer Graphics and Vision*, 15(2):113–200, 2023a. ISSN 1572-2740. doi: 10.1561/0600000107. URL http://dx.doi.org/10.1561/0600000107.

Khalid Sayood. *Introduction to data compression*. Morgan Kaufmann, 2017.

Nived Rajaraman, Jiantao Jiao, and Kannan Ramchandran. Toward a theory of tokenization in llms. *arXiv preprint arXiv:2404.08335*, 2024.

Buu Phan, Brandon Amos, Itai Gat, Marton Havasi, Matthew Muckley, and Karen Ullrich. Exact byte-level probabilities from tokenized language models for fim-tasks and model ensembles. *arXiv preprint arXiv:2410.09303*, 2024.

Peter D Grünwald, Paul MB Vitányi, et al. Algorithmic information theory. *Handbook of the Philosophy of Information*, pages 281–320, 2008.

Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

David A Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Gloria Martín. Range encoding: an algorithm for removing redundancy from a digitised message. In *Video and Data Recording Conference, Southampton, 1979*, pages 24–27, 1979.

Philip A Chou, Tom Lookabaugh, and Robert M Gray. Entropy-constrained vector quantization. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(1):31–42, 1989.

Jonas Degrave and Ira Korshunova. How we can make machine learning algorithms tunable. URL https://www.engraved.blog/how-we-can-make-machine-learning-algorithms-tunable/.

John C Platt and Alan H Barr. Constrained differential optimization for neural networks. 1988.

Ties van Rozendaal, Guillaume Sautière, and Taco S. Cohen. Lossy compression with distortion constrained optimization, 2020.

J. Ziv. On universal quantization. *IEEE Transactions on Information Theory*, 31(3):344–347, 1985.

G. J. Sullivan. Efficient scalar quantization of exponential and laplacian random variables. *IEEE Transactions on Information Theory*, 42(5):1365–1374, 1996. doi: 10.1109/18.532878.

Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

Khalid Sayood. Vector quantization. In Khalid Sayood, editor, *Introduction to Data Compression*. Morgan Kaufmann, 4 edition, 2012.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *arXiv preprint arXiv:1711.00937*, 2017.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved vqgan. *arXiv preprint arXiv:2110.04627*, 2021.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

Vivek K Goyal. Theoretical foundations of transform coding. *IEEE Signal Processing Magazine*, 18(5):9–21, 2001.

Nasir Ahmed, T. Raj Natarajan, and K. R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23:90–93, 1974.

Vivek K Goyal, Jun Zhuang, and M Veiterli. Transform coding with backward adaptive updates. *IEEE Transactions on Information Theory*, 46(4):1623–1633, 2000.

Leonardo Galteri, Lorenzo Seidenari, Marco Bertini, and Alberto Del Bimbo. Deep generative adversarial compression artifact removal. In *Proceedings of the IEEE international conference on computer vision*, pages 4826–4835, 2017.

Bahjat Kawar, Jiaming Song, Stefano Ermon, and Michael Elad. Jpeg artifact correction using denoising diffusion restoration models. *arXiv preprint arXiv:2209.11888*, 2022.

Siwei Ma, Xinfeng Zhang, Chuanmin Jia, Zhenghui Zhao, Shiqi Wang, and Shanshe Wang. Image and video compression with neural networks: A review. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(6):1683–1698, 2019.

Lyndon R Duong, Bohan Li, Cheng Chen, and Jingning Han. Multi-rate adaptive transform coding for video compression. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

B. Uria, I. Murray, and H. Larochelle. RNADE: the real-valued neural autoregressive density-estimator. In *Advances in Neural Information Processing Systems 26*, volume 26, 2013.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International conference on machine learning*, pages 2722–2730. PMLR, 2019b.

Emiel Hoogeboom, Taco Cohen, and Jakub Mikolaj Tomczak. Learning discrete distributions by dequantization. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2021b. URL https://openreview.net/forum?id=a0EpGhKt_R.

T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. PixelCNN++: A pixelcnn implementation with discretized logistic mixture likelihood and other modifications. In *International Conference on Learning Representations*, 2017.

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational Image Compression with a Scale Hyperprior. *ICLR*, 2018a.

M. Mahoney. Large Text Compression Benchmark. URL http://mattmahoney.net/dc/text.html.

B. Knoll. Cmix. URL https://www.byronknoll.com/cmix.html.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1745–1780, 1997.

F. Bellard. Lossless Data Compression with Neural Networks, 2019.

ITU-T. Recommendation ITU-T T.81: Information technology – Digital compression and coding of continuous-tone still images – Requirements and guidelines, 1992.

R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 1991.

R. Hosseini, F. Sinz, and M. Bethge. Lower bounds on the redundancy of natural images. *Vision Research*, 50(22):2213–2222, 2010.

L. Theis, R. Hosseini, and M. Bethge. Mixtures of conditional Gaussian scale mixtures applied to multiscale image representations. *PLoS ONE*, 7(7), 2012.

L. Theis and M. Bethge. Generative image modeling using spatial LSTMs. In *Advances in Neural Information Processing Systems 28*, 2015.

A. van den Oord, N. Kalchbrenner, O. Vinyals, A. Graves L. Espeholt, and K. Kavukcuoglu. Conditional Image Generation with PixelCNN Decoders. In *Advances in Neural Information Processing Systems 29*, pages 4790–4798, 2016.

N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran. Image transformer. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4055–4064, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017.

X. Chen, N. Mishra, M. Rohaninejad, and P. Abbeel. PixelSNAIL: An improved autoregressive generative model. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, pages 864–872. PMLR, 10–15 Jul 2018a.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 10215–10224, 2018.

X. Wu, K. U. Barthel, and W. Zhang. Piecewise 2D Autoregression for Predictive Image Coding. In *ICIP (3)*, pages 901–904. IEEE Computer Society, 1998.

B. Meyer and P. Tischer. Glicbawls – Grey Level Image Compression By Adaptive Weighted Least Squares. In *Data Compression Conference*, 2001.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Neural Info. Process. Sys.*, volume 31, 2018.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014a.

Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41 (8):2008–2026, 2018.

Brendan J Frey. *Bayesian networks for pattern classification, data compression, and channel coding.* Citeseer, 1998.

Chris S Wallace. Classification by minimum-message-length inference. In *International Conference on Computing and Information*, pages 72–81. Springer, 1990.

Antti Honkela and Harri Valpola. Variational learning and bits-back coding: an information-theoretic view to bayesian learning. *IEEE transactions on Neural Networks*, 15(4):800–810, 2004.

Brendan J. Frey and Geoffrey E. Hinton. Efficient stochastic source coding and an application to a bayesian network source model. *The Computer Journal*, 40(2_and_3):157–165, 1997.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.

James Townsend, Thomas Bird, Julius Kunze, and David Barber. Hilloc: Lossless image compression with hierarchical latent variable models. *arXiv preprint arXiv:1912.09953*, 2019b.

Jarek Duda. Asymmetric numeral systems, 2009.

Yibo Yang, Stephan Eckstein, Marcel Nutz, and Stephan Mandt. Estimating the rate-distortion function by wasserstein gradient descent 37th conference on neural information processing systems (neurips). In *37th Conference on Neural Information Processing Systems (NeurIPS)*, volume 36. Curran Associates, Inc., 2023b.

R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. doi: 10.1109/CVPR.2018.00068.

Saurabh Singh, Sami Abu-El-Haija, Nick Johnston, Johannes Ballé, Abhinav Shrivastava, and George Toderici. End-to-end learning of compressible features. In *ICIP*, pages 3349–3353. IEEE, 2020.

Yann Dubois, Benjamin Bloem-Reddy, Karen Ullrich, and Chris J Maddison. Lossy compression for lossless prediction. In *Neural Info. Process. Sys.*, 2021.

A. M. Eskicioglu and P. S. Fisher. Image quality measures and their performance. *IEEE Trans. Communications*, 43:2959 – 2965, 1995.

ITU-T. Recommendation ITU-T T.800.2: Methods for objective and subjective assessment of speech and video quality, 2016.

Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13 (4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, volume 2, pages 1398–1402 Vol.2, 2003. doi: 10.1109/ACSSC.2003.1292216.

Lucas Theis. What makes an image realistic? *arXiv preprint arXiv:2403.04493*, 2024.

Aaron B Wagner and Johannes Ballé. Neural networks optimally compress the sawbridge. In *2021 Data Compression Conference (DCC)*, pages 143–152. IEEE, 2021.

L. Schuchman. Dither signals and their effect on quantization noise. *IEEE Transactions on Communication Technology*, 12(4):162–165, 1964.

L. Theis and N. Yosri. Algorithms for the communication of samples. preprint, 2021. URL https://arxiv.org/abs/2110.12805.

Gergely Flamich, Marton Havasi, and José Miguel Hernández-Lobato. Compressing Images by Encoding their Latent Representations with Relative Entropy Coding. *NeurIPS*, 2020a.

E. Agustsson and L. Theis. Universally Quantized Neural Compression. In *Advances in Neural Information Processing Systems 33*, 2020.

Daniel Goc and Gergely Flamich. On channel simulation with causal rejection samplers. *arXiv preprint arXiv:2401.16579*, 2024.

Paul Cuff. Communication requirements for generating correlated random variables. In *2008 IEEE International Symposium on Information Theory*, pages 1393–1397. IEEE, 2008a.

Marton Havasi, Robert Peharz, and José Miguel Hernández-Lobato. Minimal random code learning: Getting bits back from compressed model parameters. *arXiv preprint arXiv:1810.00440*, 2018.

Sourav Chatterjee and Persi Diaconis. The sample size required in importance sampling. *The Annals of Applied Probability*, 28(2):1099–1135, 2018.

Gergely Flamich, Stratis Markou, and José Miguel Hernández-Lobato. Fast relative entropy coding with a* coding. In *International Conference on Machine Learning*, pages 6548–6577. PMLR, 2022.

Gergely Flamich, Stratis Markou, and José Miguel Hernández-Lobato. Faster relative entropy coding with greedy rejection coding. *Advances in Neural Information Processing Systems*, 36, 2024.

Gergely Flamich. Greedy poisson rejection sampling. *Advances in Neural Information Processing Systems*, 36, 2024.

Lawrence Roberts. Picture Coding using Pseudo-Random Noise. *IRE Transactions on Information Theory*, pages 145–154, 1962.

R. Zamir and M. Feder. On universal quantization by randomized uniform/lattice quantizers. *IEEE Transactions on Information Theory*, 38(2):428–436, 1992.

Mingtian Zhang, Peter Hayes, Thomas Bird, Raza Habib, and David Barber. Spread divergence. In *International Conference on Machine Learning*, pages 11106–11116. PMLR, 2020.

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational Image Compression with a Scale Hyperprior. *ICLR*, 2018b.

Jooyoung Lee, Seunghyun Cho, and Seung-Kwon Beack. Context-adaptive entropy model for end-to-end optimized image compression. In *International Conference on Learning Representations*, May 2019a.

Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken ELBO. In *International Conference on Machine Learning*, pages 159–168. PMLR, 2018.

Jun Han, Salvator Lombardo, Christopher Schroers, and Stephan Mandt. Deep generative video compression. In *Neural Information Processing Systems*, 2019.

Amirhossein Habibian, Ties van Rozendaal, Jakub M Tomczak, and Taco S Cohen. Video compression with rate-distortion autoencoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7033–7042, 2019.

Glenn Randers-Pehrson. Mng: a multiple-image format in the png family. *World Wide Web Journal*, 2(1):209–211, 1997.

Joaquim Campos, Simon Meierhans, Abdelaziz Djelouah, and Christopher Schroers. Content adaptive optimization for neural image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

L. Theis, W. Shi, A. Cunningham, and F. Huszár. Lossy Image Compression with Compressive Autoencoders. In *International Conference on Learning Representations*, 2017b.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286, 2014b.

Chris Cremer, Xuechen Li, and David Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pages 1078–1086, 2018.

Rahul Krishnan, Dawen Liang, and Matthew Hoffman. On the challenges of learning with inference networks on sparse, high-dimensional data. In *International Conference on Artificial Intelligence and Statistics*, pages 143–151, 2018.

Devon Hjelm, Russ R Salakhutdinov, Kyunghyun Cho, Nebojsa Jojic, Vince Calhoun, and Junyoung Chung. Iterative refinement of the approximate posterior for directed belief networks. In *Advances in Neural Information Processing Systems*, pages 4691–4699, 2016.

Yoon Kim, Sam Wiseman, Andrew Miller, David Sontag, and Alexander Rush. Semi-amortized variational autoencoders. In *International Conference on Machine Learning*, pages 2678–2687, 2018.

Joe Marino, Yisong Yue, and Stephan Mandt. Iterative amortized inference. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3403–3412, 2018.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. arXiv:1308.3432.

Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. *arXiv preprint arXiv:1903.05662*, 2019.

G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar. Variable rate image compression with recurrent neural networks. In *International Conference on Learning Representations*, 2016.

Eirikur Agustsson, Fabian Mentzer, Michael Tschannen, Lukas Cavigelli, Radu Timofte, Luca Benini, and Luc V Gool. Soft-to-hard vector quantization for end-to-end learning compressible representations. In *Advances in Neural Information Processing Systems*, pages 1141–1151, 2017.

Yibo Yang, Robert Bamler, and Stephan Mandt. Variational Bayesian Quantization. In *International Conference on Machine Learning*, 2020b.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Friso H Kingma, Pieter Abbeel, and Jonathan Ho. Bit-swap: Recursive bits-back coding for lossless compression with hierarchical latent variables. *arXiv preprint arXiv:1905.06845*, 2019.

N. Asuni and A. Giachetti. TESTIMAGES: A large-scale archive for testing visual devices and basic image processing algorithms (SAMPLING 1200 RGB set). In *STAG: Smart Tools and Apps for Graphics*, 2014. URL https://sourceforge.net/projects/testimages/files/OLD/OLD_SAMPLING/testimages.zip.

D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2014.

Michael D Adams. The jpeg-2000 still image compression standard. 2001.

Johannes Ballé, Sung Jin Hwang, Nick Johnston, and David Minnen. Tensorflow-compression: Data compression in tensorflow. URL https://github.com/tensorflow/compression.

Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. *VCEG-M33*, 2001.

Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. *arXiv preprint arXiv:2001.01568*, 2020.

Zhihao Duan, Ming Lu, Zhan Ma, and Fengqing Zhu. Opening the black box of learned image coders. In *2022 Picture Coding Symposium (PCS)*, pages 73–77. IEEE, 2022.

Nutan Chen, Alexej Klushyn, Francesco Ferroni, Justin Bayer, and Patrick Van Der Smagt. Learning flat latent manifolds with vaes. *arXiv preprint arXiv:2002.04881*, 2020.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

Nutan Chen, Alexej Klushyn, Richard Kurle, Xueyan Jiang, Justin Bayer, and Patrick Smagt. Metrics for deep generative models. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1550. PMLR, 2018b.

P. Cuff. Communication requirements for generating correlated random variables. In *2008 IEEE International Symposium on Information Theory*, pages 1393–1397, 2008b.

G. Flamich, M. Havasi, and J. M. Hernández-Lobato. Compressing Images by Encoding Their Latent Representations with Relative Entropy Coding, 2020b. Advances in Neural Information Processing Systems 34.

Mingtian Zhang, Peter Hayes, and David Barber. Generalization gap in amortized inference. *arXiv preprint arXiv:2205.11640*, 2022.

Guo-Hua Wang, Jiahao Li, Bin Li, and Yan Lu. Evc: Towards real-time neural image compression with mask decay. In *International Conference on Learning Representations*, 2023.

Jean Bégaint, Fabien Racapé, Simon Feltman, and Akshay Pushparaja. Compressai: a pytorch library and evaluation platform for end-to-end compression research. *arXiv preprint arXiv:2011.03029*, 2020.

CLIC. CLIC 2018 Workshop and Challenge on Learned Image Compression. https://www.compression.cc/2018/, 2018. URL http://clic.compression.cc/2018/challenge/. Accessed: 2020-04-09.

Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. Morphnet: Fast & simple resource-constrained structure learning of deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1586–1595, 2018.

O. Rippel and L. Bourdev. Real-time adaptive image compression. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

Dailan He, Yaoyan Zheng, Baocheng Sun, Yan Wang, and Hongwei Qin. Checkerboard context model for efficient learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14771–14780, 2021.

Anji Liu, Stephan Mandt, and Guy Van den Broeck. Lossless compression with probabilistic circuits. In *International Conference on Learning Representations*, 2022.

Berivan Isik, Onur G Guleryuz, Danhang Tang, Jonathan Taylor, and Philip A Chou. Sandwiched video compression: Efficiently extending the reach of standard codecs with neural wrappers. *arXiv preprint arXiv:2303.11473*, 2023.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Gary J Sullivan, Pankaj N Topiwala, and Ajay Luthra. The h. 264/avc advanced video coding standard: Overview and introduction to the fidelity range extensions. *Applications of Digital Image Processing XXVII*, 5558:454–474, 2004.

Vivienne Sze, Madhukar Budagavi, and Gary J Sullivan. High efficiency video coding (hevc). In *Integrated circuit and systems, algorithms and architectures*, volume 39, page 40. Springer, 2014.

Ties van Rozendaal, Iris A. M. Huijben, and Taco S. Cohen. Overfitting for fun and profit: Instance-adaptive data compression, 2021.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1874–1883, 2016.

Stéphane Mallat. *A Wavelet Tour of Signal Processing: The Sparse Way*. Elsevier, 2008.

Jerry Gibson. Rate distortion functions and rate distortion function lower bounds for real-world sources. *Entropy*, 19(11):604, 2017.

Matt Mahoney. Rationale for a large text compression benchmark. *Retrieved (Oct. 1st, 2021) from: http://mattmahoney.net/dc/rationale.html*, 2009.

Toby Berger and Jerry D Gibson. Lossy source coding. *IEEE Transactions on Information Theory*, 44(6):2693–2723, 1998.

J. Hayes, A. Habibi, and P. Wintz. Rate-distortion function for a gaussian source model of images (corresp.). *IEEE Transactions on Information Theory*, 16(4):507–509, 1970. doi: 10.1109/TIT.1970.1054496.

Imre Csiszár. On the computation of rate-distortion functions (corresp.). *IEEE Transactions on Information Theory*, 20(1):122–124, 1974b. doi: 10.1109/TIT.1974.1055146.

Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, Nov 2021. ISSN 1939-3539. doi: 10.1109/tpami.2020.2992934. URL http://dx.doi.org/10.1109/TPAMI.2020.2992934.

Imre Csiszár. On an extremum problem of information theory. *Studia Scientiarum Mathematicarum Hungarica*, 9, 01 1974c.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

Miguel A. Carreira-Perpinan. Mode-finding for mixtures of gaussian distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1318–1323, 2000.

Ben Poole, Sherjil Ozair, Aaron Van Den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5171–5180. PMLR, 09–15 Jun 2019. URL http://proceedings.mlr.press/v97/poole19a.html.

Sicong Huang, Alireza Makhzani, Yanshuai Cao, and Roger Grosse. Evaluating lossy compression rates of deep generative models. *International Conference on Machine Learning*, 2020.

Erwin Riegler, Günther Koliander, and Helmut Bölcskei. Rate-distortion theory for general sets and measures. *arXiv preprint arXiv:1804.08980*, 2018.

Toby Berger. *Rate distortion theory, a mathematical basis for data compression*. Prentice Hall, 1971.

Mung Chiang and Stephen Boyd. Geometric programming duals of channel capacity and rate distortion. *IEEE Transactions on Information Theory*, 50(2):245–258, 2004.

Aad W Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.

Jessica N. Howard, Stephan Mandt, Daniel Whiteson, and Yibo Yang. Foundations of a fast, data-driven, machine-learned simulator, 2021.

Zohar Jackson, César Souza, Jason Flaks, Yuxin Pan, Hereman Nicolas, and Adhish Thite. Jakobovski/free-spoken-digit-dataset: v1.0.8, August 2018. URL https://doi.org/10.5281/zenodo.1342401.

Hariharan Narayanan and Sanjoy Mitter. Sample complexity of testing the manifold hypothesis. *Advances in neural information processing systems*, 23, 2010.

Phillip Pope, Chen Zhu, Ahmed Abdelkader, Micah Goldblum, and Tom Goldstein. The intrinsic dimension of images and its impact on learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=XJk19XzGq2J.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

Victoria Kostina and Sergio Verdú. Fixed-length lossy compression in the finite blocklength regime. *IEEE Transactions on Information Theory*, 58(6):3309–3338, 2012.

Eric Lei, Hamed Hassani, and Shirin Saeedi Bidokhti. Neural estimation of the rate-distortion function with applications to operational source coding. *IEEE Journal on Selected Areas in Information Theory*, 2023a.

Robert M Gray. *Entropy and information theory*. Springer Science & Business Media, 2011.

Amir Dembo and L Kontoyiannis. Source coding, large deviations, and approximate pattern matching. *IEEE Transactions on Information Theory*, 48(6):1590–1615, 2002.

Gabriel Peyré and Marco Cuturi. Computational optimal transport: With applications to data science. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019.

Federico Bassetti, Antonella Bodini, and Eugenio Regazzini. On minimum kantorovich distance estimators. *Statistics & probability letters*, 76(12):1298–1302, 2006.

Martial Agueh and Guillaume Carlier. Barycenters in the wasserstein space. *SIAM Journal on Mathematical Analysis*, 43(2):904–924, 2011.

Philippe Rigollet and Jonathan Weed. Entropic optimal transport is maximum-likelihood deconvolution. *Comptes Rendus Mathematique*, 356(11-12):1228–1235, 2018.

Raymond J Carroll and Peter Hall. Optimal rates of convergence for deconvolving a density. *Journal of the American Statistical Association*, 83(404):1184–1186, 1988.

Bruce G Lindsay and Kathryn Roeder. Uniqueness of estimation and identifiability in mixture models. *Canadian Journal of Statistics*, 21(2):139–147, 1993.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Imre Csiszár. Information geometry and alternating minimization procedures. *Statistics and Decisions, Dedewicz*, 1:205–237, 1984.

Robert M. Gray and David L. Neuhoff. Quantization. *IEEE transactions on information theory*, 44(6):2325–2383, 1998.

Shitong Wu, Wenhao Ye, Hao Wu, Huihui Wu, Wenyi Zhang, and Bo Bai. A communication optimal transport approach to the computation of rate distortion functions. *arXiv preprint arXiv:2212.10098*, 2022.

Eric Lei, Hamed Hassani, and Shirin Saeedi Bidokhti. On a relation between the rate-distortion function and optimal transport, 2023b.

Aude Genevay, Lénaic Chizat, Francis Bach, Marco Cuturi, and Gabriel Peyré. Sample complexity of Sinkhorn divergences. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1574–1583. PMLR, 2019.

Philippe Rigollet and Austin J Stromme. On the sample complexity of entropic optimal transport. *arXiv preprint arXiv:2206.13472*, 2022.

Yuling Yan, Kaizheng Wang, and Philippe Rigollet. Learning gaussian mixtures using the wasserstein-fisher-rao gradient flow. *arXiv preprint arXiv:2301.01766*, 2023.

Lenaic Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. An interpolating distance between optimal transport and fisher–rao metrics. *Foundations of Computational Mathematics*, 18:1–44, 2018.

Lénaïc Chizat. Mean-field langevin dynamics: Exponential convergence and annealing. *arXiv preprint arXiv:2202.01009*, 2022.

Marcel Nutz. Introduction to entropic optimal transport. *Lecture notes, Columbia University*, 2021. https://www.math.columbia.edu/~mnutz/docs/EOT_lecture_notes.pdf.

Guillaume Carlier, Lénaïc Chizat, and Maxime Laborde. Lipschitz continuity of the Schrödinger map in entropic optimal transport. *arXiv preprint arXiv:2210.00225*, 2022.

Stephan Eckstein and Marcel Nutz. Convergence rates for regularized optimal transport via quantization. *arXiv preprint arXiv:2208.14391*, 2022.

Eric Lei, Hamed Hassani, and Shirin Saeedi Bidokhti. Neural estimation of the rate-distortion function for massive datasets. In *2022 IEEE International Symposium on Information Theory (ISIT)*, pages 608–613. IEEE, 2022.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

John Lafferty and Larry A Wasserman. Iterative markov chain monte carlo computation of reference priors and minimax risk. *arXiv preprint arXiv:1301.2286*, 2013.

Ao Yuan and Bertrand S Clarke. A minimally informative likelihood for decision analysis: illustration and robustness. *Canadian Journal of Statistics*, 27(3):649–665, 1999.

Andre Wibisono. Sampling as optimization in the space of measures: The langevin dynamics as a composite optimization problem. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 2093–3027. PMLR, 06–09 Jul 2018. URL https://proceedings.mlr.press/v75/wibisono18a.html.

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in Neural Information Processing Systems*, 29, 2016.

Abhishek Shetty, Raaz Dwivedi, and Lester Mackey. Distribution compression in near-linear time. *arXiv preprint arXiv:2111.07941*, 2021.

Siegfried Graf and Harald Luschgy. *Foundations of quantization for probability distributions*. Springer, 2007.

Robert M. Gray. Transportation distance, shannon information, and source coding. *GRETSI 2013 Symposium on Signal and Image Processing*, 2013. URL https://ee.stanford.edu/~gray/gretsi.pdf.

L. Theis and E. Agustsson. On the advantages of stochastic encoders. arXiv:2102.09270, 2021. URL https://arxiv.org/abs/2102.09270.

Zeyu Yan, Fei Wen, Rendong Ying, Chao Ma, and Peilin Liu. On perceptual lossy compression: The cost of perceptual reconstruction and an optimal training framework. In *International Conference on Machine Learning*, pages 11682–11692. PMLR, 2021.

Michael Chang, Tom Griffiths, and Sergey Levine. Object representations as fixed points: Training iterative refinement algorithms with implicit differentiation. *Advances in Neural Information Processing Systems*, 35:32694–32708, 2022.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

Emilien Dupont, Adam Goliński, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. Coin: Compression with implicit neural representations. *arXiv preprint arXiv:2103.03123*, 2021.

Emilien Dupont, Hrushikesh Loya, Milad Alizadeh, Adam Golinski, Yee Whye Teh, and Arnaud Doucet. Coin++: Data agnostic neural compression. *arXiv preprint arXiv:2201.12904*, 1(2):4, 2022.

Ishit Mehta, Michaël Gharbi, Connelly Barnes, Eli Shechtman, Ravi Ramamoorthi, and Manmohan Chandraker. Modulated periodic activations for generalizable local functional representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14214–14223, 2021.

Tuan Pham, Yibo Yang, and Stephan Mandt. Autoencoding implicit neural representations for image compression. In *ICML 2023 Workshop Neural Compression: From Information Theory to Applications*, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Rui Shu and Stefano Ermon. Bit prioritization in variational autoencoders via progressive coding. In *ICML*, pages 20141–20155, 2022.

Georgios Batzolis, Jan Stanczuk, and Carola-Bibiane Schönlieb. Your diffusion model secretly knows the dimension of the data manifold. *arXiv e-prints*, pages arXiv–2212, 2022.

Balázs Kégl. Intrinsic dimension estimation using packing numbers. *Advances in neural information processing systems*, 15, 2002.

Tsutomu Kawabata and Amir Dembo. The rate-distortion dimension of sets and measures. *IEEE transactions on information theory*, 40(5):1564–1572, 1994.

Thomas S Ferguson. *A course in large sample theory*. Routledge, 2017.

Miguel A. Carreira-Perpinan. How many modes can a Gaussian mixture have, 2020. URL https://faculty.ucmerced.edu/mcarreira-perpinan/research/GMmodes.html.

Jasper C. H. Lee, Jerry Li, Christopher Musco, Jeff M. Phillips, and Wai Ming Tai. Finding the mode of a kernel density estimate, 2019b.

Miguel A. Carreira-Perpinan. Gaussian mean-shift is an em algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):767–776, 2007. doi: 10.1109/TPAMI.2007.1057.

Paul Milgrom and Ilya Segal. Envelope theorems for arbitrary choice sets. *Econometrica*, 70 (2):583–601, 2002.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2019.

Erhan Çinlar. *Probability and stochastics*, volume 261. Springer, 2011.

Gerald B Folland. *Real analysis: modern techniques and their applications*, volume 40. John Wiley & Sons, 1999.

MJ Beal and Z Ghahramani. The variational bayesian em algorithm for incomplete data: with application to scoring graphical model structures. *Bayesian statistics*, 7(453-464):210, 2003b.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1):1–22, 1977.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37:183–233, 1999.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

Michael Irwin Jordan. *Learning in graphical models*. MIT press, 1999.

Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic gradient descent. In *International Conference on Learning Representations*, 2015.

Roger B Grosse, Zoubin Ghahramani, and Ryan P Adams. Sandwiching the marginal likelihood using bidirectional monte carlo. *arXiv preprint arXiv:1511.02543*, 2015.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Appendix A

For Chapter 4, "Inference Optimization"

A.1 Stochastic Annealing

Here we provide conceptual illustrations of our stochastic annealing idea on a simple example.

Consider the following scalar optimization problem over integers,

$$\underset{z\in\mathbb{Z}}{\text{minimize}} \quad f(z) = z^2$$

Following our stochastic annealing method in Section. 4.3.2, we let $x \in \mathbb{R}$ be a continuous proxy variable, and $r \in \{(1,0),(0,1)\}$ be the one-hot vector of stochastic rounding direction. We let r follow a Bernoulli distribution with temperature $\tau > 0$,

$$q_\tau(r|x) = \begin{cases} \exp\big\{-\psi\big(x - \lfloor x \rfloor\big)/\tau\big\}/C(x,\tau) & \text{if } r = (1,0) \\ \exp\big\{-\psi\big(\lceil x \rceil - x\big)/\tau\big\}/C(x,\tau) & \text{if } r = (0,1) \end{cases}$$

where

$$C(x,\tau) = \exp\{-\psi(x - \lfloor x \rfloor)/\tau\} + \exp\{-\psi(\lceil x \rceil - x)/\tau\}$$

is the normalizing constant.

The stochastic optimization problem for SGA is then

$$\begin{aligned}\underset{x\in\mathbb{R}}{\text{minimize}} \quad \ell_\tau(x) &:= \mathbb{E}_{q_\tau(r|x)}[f(r \cdot (\lfloor x \rfloor, \lceil x \rceil)] \\ &= \frac{\exp\{-\psi(x - \lfloor x \rfloor)/\tau\}}{C(x,\tau)} f(\lfloor x \rfloor) + \frac{\exp\{-\psi(\lceil x \rceil - x)/\tau\}}{C(x,\tau)} f(\lceil x \rceil)\end{aligned}$$

Below we plot the rounding probability $q_\tau(r = (0,1)|x)$ and the SGA objective ℓ_τ as a function of x, at various temperatures.

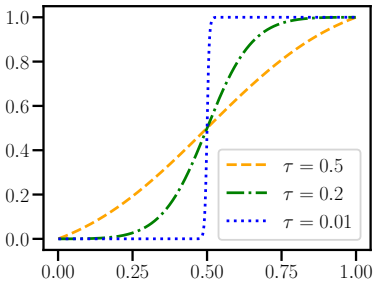


Figure A.1: Illustration of the probability $q_\tau(r = (0,1)|x)$ of rounding *up*, on the interval $(0,1)$, at various temperatures.

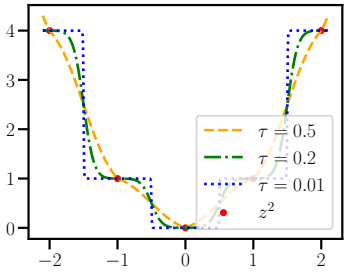


Figure A.2: Graphs of stochastic annealing objective ℓ_τ at various temperatures. ℓ_τ interpolates the discrete optimization problem between integers, and approaches $\text{round}(x)^2$ as $\tau \to 0$.

A.2 Model Architecture and Training

As mentioned in the main text, the *Base Hyperprior* model uses the same architecture as in Table 1 of Minnen et al. [2018] except without the "Context Prediction" and "Entropy Parameters" components (this model was referred to as "Mean & Scale Hyperprior" in this

work). The model is mostly already implemented in `bmshj2018.py`[1] from Ballé et al., Ballé et al. [2018b], and we modified their implementation to double the number of output channels of the `HyperSynthesisTransform` to predict both the mean and (log) scale of the conditional prior $p(\mathbf{y}|\mathbf{z})$ over the latents $\mathbf{y}$.

As mentioned in Section 4.3.3 and 4.4, lossy bits-back modifies the above *Base Hyperprior* as follows:

1. The number of output channels of the hyper inference network is doubled to compute both the mean and diagonal (log) variance of Gaussian $q(\mathbf{z}|\mathbf{x})$;

2. The hyperprior $p(\mathbf{z})$ is no longer restricted to the form of a flexible density model convolved with the uniform distribution on $[-0.5, 0.5]$; instead it simply uses the flexible density, as described in the Appendix of Ballé et al. [2018b].

We trained our models (both the *Base Hyperprior* and the bits-back variant) on CLIC-2018 [2] images using minibatches of eight 256×256 randomly-cropped image patches, for $\lambda \in \{0.001, 0.0025, 0.005, 0.01, 0.02, 0.04, 0.08\}$. Following Ballé et al. [2018b], we found that increasing the number of latent channels helps avoid the "bottleneck" effect and improve rate-distortion performance at higher rates, and increased the number of latent channels from 192 to 256 for models trained with $\lambda = 0.04$ and 0.08. All models were trained for 2 million steps, except the ones with $\lambda = 0.001$ which were trained for 1 million steps, and $\lambda = 0.08$ which were trained for 3 million steps. Our code and pre-trained models can be found at `https://github.com/mandt-lab/improving-inference-for-neural-image-compression`.

[1] `https://github.com/tensorflow/compression/blob/master/examples/bmshj2018.py`
[2] `https://www.compression.cc/2018/challenge/`

A.3 Hyper-Parameters of Various Methods Considered

Given a pre-trained model (see above) and image(s) $\mathbf{x}$ to be compressed, our experiments in Section 4.4 explored hybrid amortized-iterative inference to improve compression performance. Below we provide hyper-parameters of each method considered:

Stochastic Gumbel Annealing. In all experiments using SGA (including the *SGA+BB* experiments, which optimized over $\hat{\mathbf{y}}$ using SGA), we used the Adam optimizer with an initial learning rate of 0.005, and an exponentially decaying temperature schedule $\tau(t) = \min(\exp\{-ct\}, 0.5)$, where t is the iteration number, and $c > 0$ controls the speed of the decay. We found that lower c generally increases the number of iterations needed for convergence, but can also yield slightly better solutions; in all of our experiments we set $c = 0.001$, and obtained good convergence with 2000 iterations.

Temperature Annealing Schedule for SGA. We initially used a naive temperature schedule, $\tau(t) = \tau_0 \exp\{-ct\}$ for SGA, where τ_0 is the initial temperature (typically set to 0.5 to simulate soft quantization), and found that aggressive annealing with a large decay factor c can lead to suboptimal solutions, as seen with $c = 0.002$ or $c = 0.001$ in the left subplot of Figure A.3. We found that we can overcome the suboptimality from fast annealing by an initial stage of optimization at a fixed temperature before annealing: in the initial stage, we fix the temperature to some relatively high value τ_0 (to simulate soft discretization), and run the optimization for t_0 steps such that the R-D objective roughly converges. We demonstrate this in the right subplot of Figure A.3, where we modify the naive schedule to $\tau(t) = \min\{\tau_0, \tau_0 \exp\{-c(t - t_0)\}\}$, so that SGA roughly converges after $t_0 = 700$ steps at temperature $\tau_0 = 0.5$ before annealing. As can be seen, the results of SGA are robust to different choices of decay factor c, with $c = 0.002$ and $c = 0.001$ giving comparably good results to $c = 0.0005$ but with faster convergence.

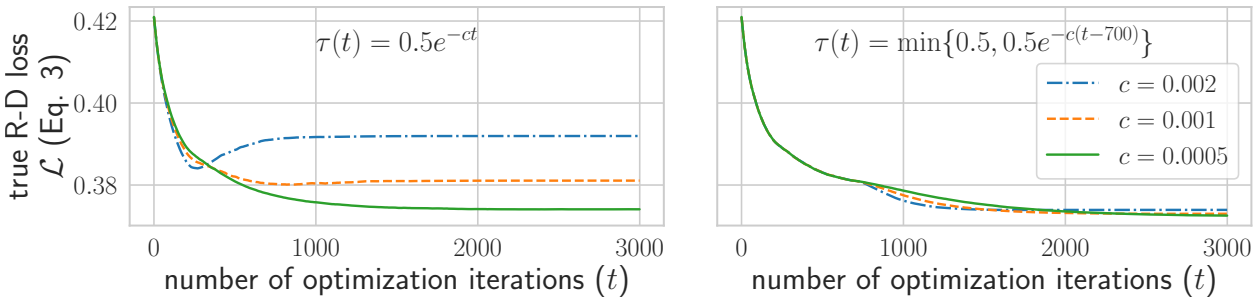


Figure A.3: Comparing different annealing schedules for SGA (Section 4.3.2).

Lossy bits-back. In the *SGA+BB* experiments, line 2 of the `encode` subroutine of Algorithm 1 performs joint optimization w.r.t. $\mu_{\mathbf{y}}$, $\mu_{\mathbf{z}}$, and $\sigma^2_{\mathbf{z}}$ with Black-box Variational Inference (BBVI), using the reparameterization trick to differentiate through Gumbel-softmax samples for $\mu_{\mathbf{y}}$, and Gaussian samples for $(\mu_{\mathbf{z}}, \sigma^2_{\mathbf{z}})$. We used Adam with an initial learning rate of 0.005, and ran 2000 stochastic gradient descent iterations; the optimization w.r.t. $\mu_{\mathbf{y}}$ (SGA) used the same temperature annealing schedule as above. The `reproducible_BBVI` subroutine of Algorithm 1 used Adam with an initial learning rate of 0.003 and 2000 stochastic gradient descent iterations; we used the same random seed for the encoder and decoder for simplicity, although a hash of $\hat{\mathbf{y}}$ can also be used instead.

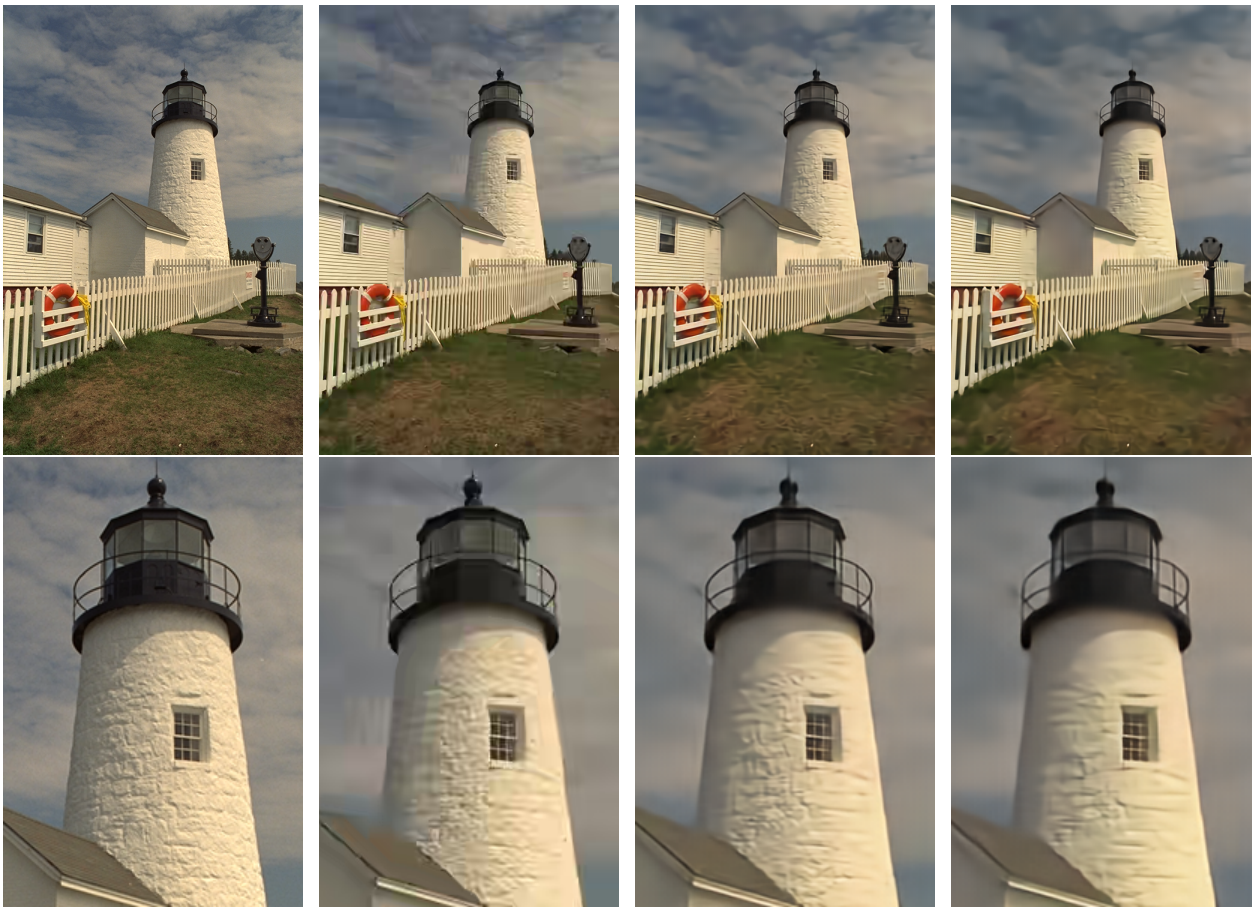
Alternative Discretization Methods In Figure 4.3 and ablation studies of Section 4.4, all alternative discretization methods were optimized with Adam for 2000 iterations to compare with *SGA*. The initial learning rate was 0.005 for *MAP*, *Uniform Noise*, and *Deterministic Annealing*, and 0.0001 for *STE*. All methods were tuned on a best-effort basis to ensure convergence, except that *STE* consistently encountered convergence issues even with a tiny learning rate (see [Yin et al., 2019]). The rate-distortion results for *MAP* and *STE* were calculated with early stopping (i.e., using the intermediate $(\hat{\mathbf{y}}, \hat{\mathbf{z}})$ with the lowest true rate-distortion objective during optimization), just to give them a fair chance. Lastly, the comparisons in Figure 4.3 used the *Base Hyperprior* model trained with $\lambda = 0.0025$.

A.4 Additional Results

On Kodak, we achieve the following BD rate savings: 17% for *SGA+BB* and 15% for *SGA* relative to *Base Hyperprior*; 17% for *SGA+BB* and 15% for *SGA* relative to BPG. On Tecnick, we achieve the following BD rate savings: 20% for *SGA+BB* and 19% for *SGA* relative to *Base Hyperprior*; 22% for *SGA+BB* and 21% for *SGA* relative to BPG.

Our experiments were conducted on a Titan RTX GPU with 24GB RAM; we observed about a 100× slowdown from our proposed iterative inference methods (compared to standard encoder network prediction), similar to what has been reported in [Campos et al., 2019]. Note that our proposed standalone variant [M1] with SGA changes only compression and does not affect *decompression* speed (which is more relevant, e.g., for images on a website with many visitors).

Below we provide an additional qualitative comparison on the Kodak [1993] dataset, and report detailed rate-distortion performance on the Tecnick [Asuni and Giachetti, 2014] dataset.



(a) Original image (kodim19 from Kodak [1993])

(b) BPG 4:4:4; 0.143 BPP (PSNR = 29.3)

(c) Proposed (SGA); 0.142 BPP (PSNR = 29.8)

(d) [Minnen et al., 2018]; 0.130 BPP (PSNR = 28.5)

Figure A.4: Qualitative comparison of lossy compression performance on an image from the Kodak [1993] dataset. Figures in the bottom row focus on the same cropped region of images in the top row. Our method (c; [M1] in Table 4.1) significantly boosts the visual quality of the *Base Hyperprior* method (d; [M3] in Table 4.1) at similar bit rates. Unlike the learning-based methods (subfigures c,d), the classical codec BPG (subfigure b) introduces blocking and geometric artifacts near the rooftop, and ringing artifacts around the contour of the lighthouse.

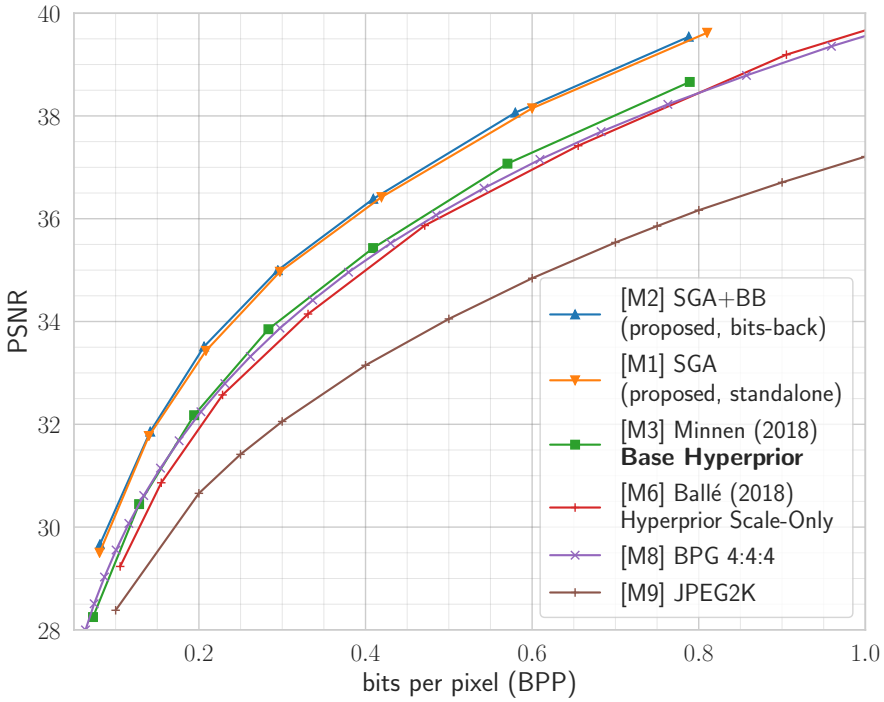


Figure A.5: Rate-distortion performance comparisons on the Tecnick [Asuni and Giachetti, 2014] dataset against existing baselines. Image quality measured in Peak Signal-to-Noise Ratio (PSNR) in RGB; higher values are better.

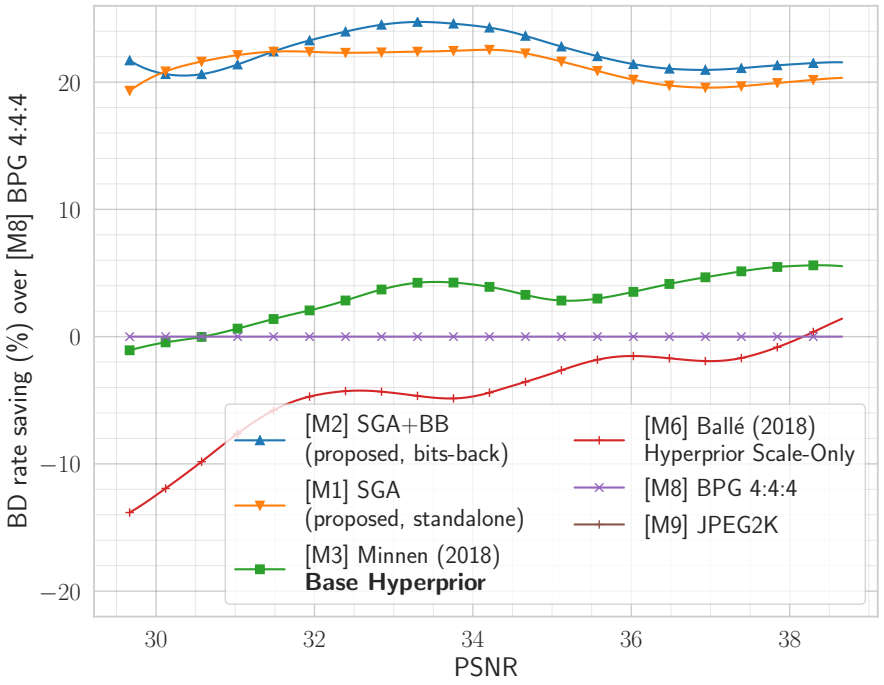


Figure A.6: BD rate savings (%) relative to BPG as a function of PSNR, computed from R-D curves (Figure A.5) on Tecnick. Higher values are better.

Appendix B

For Chapter 5, "Asymmetrically-Powered Neural Compression for Decoding Efficiency"

B.1 More Details on the Two-Layer Synthesis with Residual Connection

Fig. B.1 illustrates the proposed two-layer architecture with a residual connection, where $\mathbf{a} = \texttt{conv\_1}(\mathbf{z})$ denotes the output of the first transposed conv layer. We implement the residual connection (the lower computation path in the figure) with another transposed convolution layer `conv_res`, using the same configuration (stride, kernel size, etc.) as `conv_1`. In our main experiments we use $k_1 = 13, s_1 = 8, k_2 = 5, s_2 = 2$ and $N = 12$.

The residual connection $\mathbf{r}$ is inspired by its success in recent NTC architectures [Cheng et al., 2020, He et al., 2022], and can also be interpreted as a data-dependent and spatially-

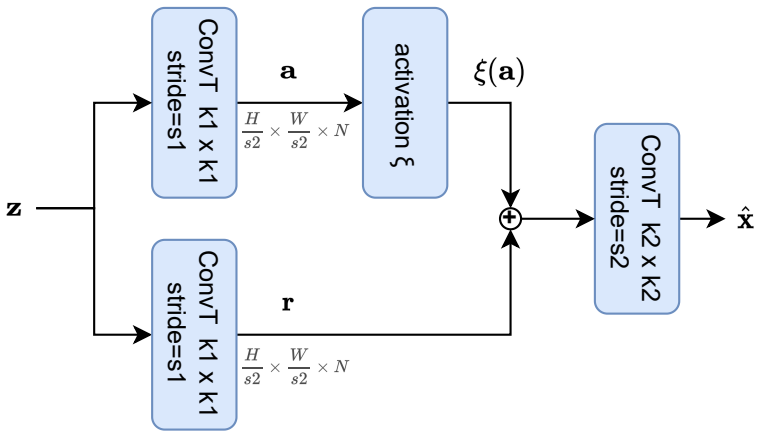


Figure B.1: Diagram of the proposed two-layer synthesis transform.

varying bias term that modulates the nonlinear activation $\xi(\mathbf{a})$. At lower bit-rates, we found employing the residual connection to be more parameter- and compute-efficient than a simple composition of two transposed convolution layers without the residual connection.

In further experiments, we re-trained models with "mixed quantization" [Minnen and Singh, 2020] instead of additive uniform noise as in the main paper, and found that with comparable decoding complexity, a simple two-layer architecture with $N = 24$ hidden channels (and no residual connection) in fact slightly outperforms the one with residual connection and $N = 12$, while keeping all other hyperparameters the same.

B.2 Theoretical Result

In the following, we derive the decomposition of the R-D cost of neural lossy compression. To lighten notation, we use non-bold letters (x, z instead of $\mathbf{x}, \mathbf{z}$), and adopt the general setting where the latent space $\mathcal{Z}$ is a Polish space (which includes, among many other examples, the Euclidean space $\mathbb{R}^{|\mathcal{Z}|}$ commonly used for continuous latent variables, or the set of lattice points $\mathbb{Z}^{|\mathcal{Z}|}$ in nonlinear transform coding), and P_Z is a prior probability measure. We present results in terms of measures for generality, but for readers unfamiliar with measure theory it

is harmless to focus on the common case where P_Z admits a density $p(z)$ (denoted $p(\mathbf{z})$ in the main paper), s.t., $P_Z(dz) = p(z)dz$; in the discrete case, $p(z)$ is a PMF, and the integral (w.r.t. the counting measure on $\mathcal{Z}$) reduces to a sum. Similarly, $Q_{Z|X}$ is family of probability measures, such that for each value of x it defines a conditional distribution $Q_{Z|X=x}$, which may admit a density $q(z|x)$.

A (learned) lossy compression codec consists of a prior distribution P_Z, a stochastic encoding transform $Q_{Z|X}$, and a deterministic decoding transform $g : \mathcal{Z} \to \hat{\mathcal{X}}$. Suppose relative entropy coding [Cuff, 2008b, Theis and Yosri, 2021, Flamich et al., 2020b] operates with minimal overhead, i.e., a sample from $Q_{Z|X=x}$ can be transmitted under P_Z with a bit-rate close to $KL(Q_{Z|X=x}\|P_Z)$ (which may require us to perform block coding), then given a data realization x, the rate-distortion compression cost is, on average,

$$\mathcal{L}(Q_{Z|X}, g, P_Z, x) := \lambda \mathbb{E}_{z\sim Q_{Z|X=x}}[\rho(x, g(z))] + KL(Q_{Z|X=x}\|P_Z), \tag{B.1}$$

where $\rho : \mathcal{X} \times \hat{\mathcal{X}} \to [0, \infty)$ is the distortion function, and $\lambda \geq 0$ is a fixed hyperparameter trading off between rate and distortion, and both ρ and λ are specified by the lossy compression problem in advance.

By Lemma 8.5 of [Gray, 2011], this compression cost admits a similar decomposition to the negative ELBO,

$$\mathcal{L}(Q_{Z|X}, g, P_Z, x) = -\log \Gamma_{g,P_Z}(x) + KL(Q_{Z|X=x}\|P_{Z|X=x}), \tag{B.2}$$

where $P_{Z|X}$ denotes the Markov kernel (transitional distribution) defined by

$$P_{Z|X=x}(dz) := \frac{e^{-\lambda\rho(x,g(z))}P_Z(dz)}{\Gamma_{g,P_Z}(x)}, \tag{B.3}$$

and the normalizing constant is

$$\Gamma_{g,P_Z}(x) := \int_{\mathcal{Z}} e^{-\lambda\rho(x,g(z))} P_Z(dz). \tag{B.4}$$

As in variational Bayesian inference, the normalizing constant has the interpretation of a marginal log-likelihood specified by the prior P_Z and model g. We note that the definition of $P_{Z|X}$ depends on g and P_Z, but leave this out to lighten notation. Eq. B.2 together with the non-negativity of KL divergence imply that $P_{Z|X}$ is the optimal channel ("inference distribution") for reverse channel coding, under which the compression cost equals:

$$\min_{Q_{Z|X=x}} \lambda\mathbb{E}_{z\sim Q_{Z|X=x}}[\rho(x,g(z))] + KL(Q_{Z|X=x}\|P_Z) = -\log\Gamma_{g,P_Z}(x) \tag{B.5}$$

Let P_X be the data distribution. Taking the expected value of Eq. B.1 w.r.t. $x \sim P_X$ gives the population-level compression cost, which can then be rewritten as follows

$$\begin{aligned}
\mathcal{L}(Q_{Z|X}, g, P_Z) &:= \lambda\mathbb{E}_{P_X Q_{Z|X}}[\rho(X,g(Z))] + \mathbb{E}_{x\sim P_X}[KL(Q_{Z|X=x}\|P_Z)] && \text{(B.6)}\\
&= \mathbb{E}_{x\sim P_X}[-\log\Gamma_{g,P_Z}(x)] + \mathbb{E}_{x\sim P_X}[KL(Q_{Z|X=x}\|P_{Z|X=x})] && \text{(B.7)}\\
&= \inf_{P'_Z,\omega} \mathbb{E}_{x\sim P_X}[-\log\Gamma_{\omega,P'_Z}(x)] + \left(\mathbb{E}_{x\sim P_X}[-\log\Gamma_{g,P_Z}(x)] - \inf_{P'_Z,\omega} \mathbb{E}_{x\sim P_X}[-\log\Gamma_{\omega,P'_Z}(x)]\right) && \text{(B.8)}\\
&\quad + \mathbb{E}_{x\sim P_X}[KL(Q_{Z|X=x}\|P_{Z|X=x})] && \text{(B.9)}\\
&= \inf_{\omega\in\mathcal{G}} F_\omega(\lambda) + \underbrace{\left(\mathbb{E}_{x\sim P_X}[-\log\Gamma_{g,P_Z}(x)] - \inf_{\omega\in\mathcal{G}} F_\omega(\lambda)\right)}_{\text{modeling gap}} + \underbrace{\mathbb{E}_{x\sim P_X}[KL(Q_{Z|X=x}\|P_{Z|X=x})]}_{\text{inference gap}}, && \text{(B.10)}
\end{aligned}$$

where for any choice of decoder function $\omega \in \mathcal{G}$, we define

$$F_\omega(\lambda) := \inf_{Q_{Z|X}} I(X;Z) + \lambda\mathbb{E}[\rho(X,\omega(Z))] \tag{B.11}$$

as the R-axis intercept of the line tangent to the ω-dependent rate-distortion function [Yang and Mandt, 2022] [1] with slope $-\lambda$. The last equation follows from Eq. B.5 and the following calculation

$$\inf_{P'_Z,\omega} \mathbb{E}_{x\sim P_X}[-\log \Gamma_{\omega,P'_Z}(x)] \tag{B.12}$$

$$= \inf_{P'_Z,\omega} \inf_{Q_{Z|X}} \lambda \mathbb{E}_{P_X Q_{Z|X}}[\rho(X,\omega(Z))] + \mathbb{E}_{x\sim P_X}[KL(Q_{Z|X=x}\|P'_Z)] \tag{B.13}$$

$$= \inf_{\omega} \inf_{Q_{Z|X}} \inf_{P'_Z} \lambda \mathbb{E}_{P_X Q_{Z|X}}[\rho(X,\omega(Z))] + \mathbb{E}_{x\sim P_X}[KL(Q_{Z|X=x}\|P'_Z)] \tag{B.14}$$

$$= \inf_{\omega} \inf_{Q_{Z|X}} \lambda \mathbb{E}_{P_X Q_{Z|X}}[\rho(X,\omega(Z))] + \inf_{P'_Z} \mathbb{E}_{x\sim P_X}[KL(Q_{Z|X=x}\|P'_Z)] \tag{B.15}$$

$$= \inf_{\omega} \inf_{Q_{Z|X}} \lambda \mathbb{E}_{P_X Q_{Z|X}}[\rho(X,\omega(Z))] + I(P_X Q_{Z|X}) \tag{B.16}$$

$$= \inf_{\omega} F_\omega(\lambda) \tag{B.17}$$

To summarize, we have broken down the R-D cost for a data source P_X into three terms,

$$\mathcal{L}(Q_{Z|X},\omega,P_Z) = \inf_{\omega\in\mathcal{G}} F_\omega(\lambda) + \underbrace{\left(\mathbb{E}_{x\sim P_X}[-\log \Gamma_{g,P_Z}(x)] - \inf_{\omega\in\mathcal{G}} F_\omega(\lambda)\right)}_{\text{modeling gap}} + \underbrace{\mathbb{E}_{x\sim P_X}[KL(Q_{Z|X=x}\|P_{Z|X=x})]}_{\text{inference gap}}. \tag{B.18}$$

- The first term (which we dentoed by the shorthand "$\mathcal{F}(\mathcal{G})$" in the main text) represents the fundamentally irreducible cost of compression determined by the source P_X and the family $\mathcal{G}$ of decoding transforms used. This is the information-theoretically optimal cost of compression within the transform family $\mathcal{G}$. If we let $F(\lambda) := \inf_{Q_{\hat{X}|X}} I(X;\hat{X}) + \lambda\mathbb{E}[\rho(X,\hat{X})]$ be the optimal Lagrangian associated with the R-D function of P_X, then it can be shown that [Yang and Mandt, 2022] $F(\lambda) \le F_g(\lambda)$. When the latent space $\mathcal{Z}$ and the transform family $\mathcal{G}$ are sufficiently large, it holds that $F(\lambda) = \inf_g F_g(\lambda)$, i.e., the first term of the R-D cost is determined solely by the rate-distortion function of the

[1] The g-dependent distortion function [Yang and Mandt, 2022] is defined as $R_g(D) := \inf_{Q_{Z|X}:\mathbb{E}[\rho(X,g(Z))]\le D} I(X;Z)$.

source distribution P_X (and distortion function ρ), which is the lossy analogue of the Shannon entropy.

- The second term represents the excess cost of doing compression with a *particular* transform g and prior P_Z compared to the *best possible* transform and prior, while always operating with the optimal channel (Eq. B.2) in each case. Note that this term only depends on the modeling choices of g and P_Z, and does not depend on the encoding/inference distribution $Q_{Z|X}$; therefore we call it the *modeling gap*. It is due largely to imperfect model training/optimization, and/or a mismatch between the training data and the target data P_X (which may not be the same).

- The third term represents the overhead of compression caused by a (potentially) sub-optimal encoding/inference distribution $Q_{Z|X}$, given a particular model g and P_Z. This overhead can be eliminated by using the optimal channel $P_{Z|X}$ given in Eq. B.2 (which depends on g and P_Z). Therefore we call this term the *inference gap*.

Remarks The decomposition of the lossy compression cost has a natural parallel to that of lossless compression under a latent variable model.

Consider a latent variable model $(p_\theta(z), p_\theta(x|z))$ with parameter vector θ, which defines a model of the marginal data density $p_\theta(x) := \int_{\mathcal{Z}} p_\theta(x|z)p_\theta(z)dz$. The cost of lossless compression under ideal bits-back coding [Frey, 1998, Townsend et al., 2019a] is equal to the negative ELBO, and admits a similar decomposition [Zhang et al., 2022]:

$$\mathbb{E}_{x\sim P_X}[\mathbb{E}_{z\sim q(z|x)}[-\log p_\theta(x|z)] + KL(q(z|x)\|p_\theta(z))] \tag{B.19}$$

$$=\mathbb{E}_{x\sim P_X}[-\log p_\theta(x)] + \mathbb{E}_{x\sim P_X}[KL(q(z|x)\|p_\theta(z|x))] \tag{B.20}$$

$$= \underbrace{H[P_X]}_{\text{data entropy}} + \underbrace{KL(P_X\|p_\theta(x))}_{\text{modeling gap}} + \underbrace{\mathbb{E}_{x\sim P_X}[KL(q(z|x)\|p_\theta(z|x))]}_{\text{inference gap}} \tag{B.21}$$

Again, we have decomposed the compression cost into a first term that represents the intrinsic compressibility of the data, a second term that depends entirely on the choice of the model, and a third overhead term from using a sub-optimal inference distribution $q(z|x)$ and which can be eliminated using the optimal inference distribution $p_\theta(z|x) \propto p_\theta(x|z)p_\theta(z)$ (the Bayesian posterior) given each choice of our model.

B.3 Implementation and Reproducibility Details

Our models are implemented in tensorflow using the `tensorflow-compression`[2] library. We implemented the Mean-Scale Hyperprior model based on the open source code [3] and architecture details from Johnston et al. [2019]. We borrowed the ELIC [He et al., 2022] transforms from the VCT repo [4].

Our experiments were run on Titan RTX GPUs. All the models were trained with the Adam optimizer following standard procedure (e.g., in [Minnen et al., 2018]) for a maximum of 2 million steps. We use an initial learning rate of $1e-4$, then decay it to $1e-5$ towards the end of training. For each model architecture, we trained separate models for $\lambda \in \{0.00125, 0.0025, 0.005, 0.01, 0.02, 0.04, 0.08\}$. For SGA [Yang et al., 2020a], we use similar hyperparameters as in [Yang et al., 2020a], using Adam optimizer, learning rate $5e-3$, and a temperature schedule of $\tau(t) = 0.5 \exp\{-0.0005(\max(0, t-200))\}$ for 3000 gradient steps.

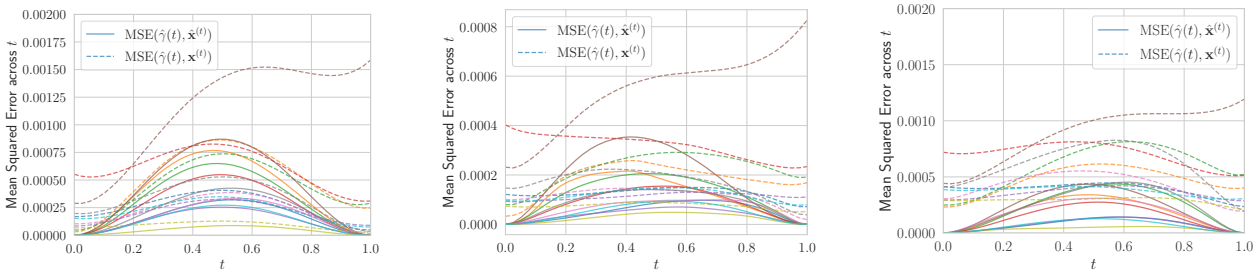
[2] https://github.com/tensorflow/compression
[3] https://github.com/tensorflow/compression/blob/master/models/bmshj2018.py
[4] https://github.com/google-research/google-research/blob/master/vct/src/elic.py

B.4 Additional Results

Results on the reconstruction manifold.

We train three popular NTC architectures [Ballé et al., 2018a, Minnen et al., 2018, Minnen and Singh, 2020] for MSE distortion with $\lambda = 0.08$ and observe similar results when traversing the manifold of reconstructed images. We use random pairs of image crops from COCO [Lin et al., 2014] to define the start and end points of latent traversal (see Sec. 5.3.1); we use the same random seed across the three different architectures. All the images are scaled to $[-0.5, 0.5]$.

MSEs between trajectories In Fig. B.2 shows the distance between the resulting trajectory of decoded curve $\hat{\gamma}(t)$ and two kinds of straight paths, interpolating between reconstructions $\hat{\mathbf{x}}^{(t)} := (1-t)\hat{\mathbf{x}}^{(0)} + t\hat{\mathbf{x}}^{(1)}$ or ground truth images $\mathbf{x}^{(t)} := (1-t)\mathbf{x}^{(0)} + t\mathbf{x}^{(1)}$. See detailed discussions in Sec. 5.3.1.



(a) Factorized Prior [Ballé et al., 2018a].

(b) Mean-Scale Hyperprior [Minnen et al., 2018].

(c) Channel-wise Autoregressive [Minnen and Singh, 2020].

Figure B.2: The distance from the trajectory of decoded curve $\hat{\gamma}(t)$ to the straight path between end-point reconstructions $\hat{\mathbf{x}}^{(t)} := (1-t)\hat{\mathbf{x}}^{(0)} + t\hat{\mathbf{x}}^{(1)}$, and to the straight path between ground truth images $\hat{\mathbf{x}}^{(t)} := (1-t)\mathbf{x}^{(0)} + t\mathbf{x}^{(1)}$.

Quantifying the curvature of decoded curves of reconstructions. Additionally, we quantify how much the curves of reconstructed images deviate from straight paths by computing the curve lengths. Recall given two tensors of latent coefficients $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$ (obtained

by passing two images $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$ through the analysis transform), we let $\gamma : [0, 1] \to \mathcal{Z}$ be the straight line in the latent space defined by their convex combination, i.e., $\gamma(t) := (1-t)\mathbf{z}^{(0)} + t\mathbf{z}^{(1)}$. The curve of reconstructions is then defined by $\hat{\gamma}(t) := g(\gamma(t))$, with end-points $\hat{\mathbf{x}}^{(0)} := g(\mathbf{z}^{(0)})$ and $\hat{\mathbf{x}}^{(1)} := g(\mathbf{z}^{(1)})$.

Following Chen et al. [2018b], the curve length of $\hat{\gamma}$ is given by

$$L(\hat{\gamma}) := \int_0^1 \|\frac{\partial g(\gamma(t))}{\partial t}\| \mathrm{d}t = \int_0^1 \|\frac{\partial g(\gamma(t))}{\partial \gamma(t)} \frac{\partial \gamma(t)}{\partial t}\| \mathrm{d}t = \int_0^1 \|\mathbf{J}_t \mathbf{v}\| \mathrm{d}t = \int_0^1 \sqrt{\mathbf{v}^T {\mathbf{J}_t}^T \mathbf{J}_t \mathbf{v}}\ \mathrm{d}t \tag{B.22}$$

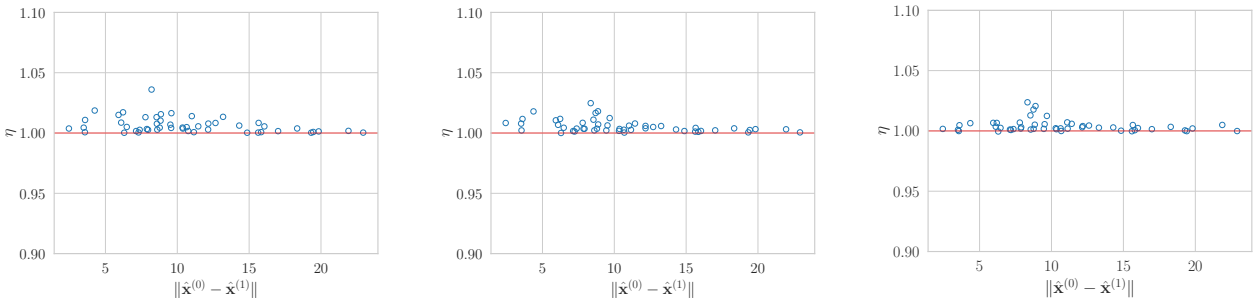
where $\mathbf{J}_t \in \mathbb{R}^{|\mathcal{X}| \times |\mathcal{Z}|}$ is the Jacobian of the synthesis transform evaluated at $\mathbf{z}^{(t)} = \gamma(t)$, and $\mathbf{v} = \frac{\partial \gamma(t)}{\partial t} = \mathbf{z}^{(1)} - \mathbf{z}^{(0)}$ is the (constant) curve velocity. As in [Chen et al., 2018b], we compute this integral approximately with a Riemann sum.

The shortest path (in Euclidean geometry) between the two end-points of $\hat{\gamma}$ is given simply by the linear interpolation $(1-t)\hat{\mathbf{x}}^{(0)} + t\hat{\mathbf{x}}^{(1)}$, with a distance of $\|\hat{\mathbf{x}}^{(1)} - \hat{\mathbf{x}}^{(0)}\|$. Therefore, we define the curve-to-shortest-path length ratio

$$\eta := \frac{L(\hat{\gamma})}{\|\hat{\mathbf{x}}^{(1)} - \hat{\mathbf{x}}^{(0)}\|} \tag{B.23}$$

as a measure of how much the curve $\hat{\gamma}$ deviates from a straight path, with $\eta = 1$ indicating a completely straight line.

We compute the curve-to-shortest-path-length ratio η on 50 randomly chosen image pairs in three NTC architectures [Ballé et al., 2018a, Minnen et al., 2018, Minnen and Singh, 2020]. We use random 16×16 image crops (the results are similar for larger images) from COCO [Lin et al., 2014]. We compute the curve length integral in Eq. B.22 via a Riemann sum, $\frac{1}{T} \sum_{t_i} \|\mathbf{J}_{t_i}(\mathbf{z}^{(1)} - \mathbf{z}^{(0)})\|$, with $T = 100$. Fig. B.3 plots the resulting η values against the



(a) Factorized Prior [Ballé et al., 2018a].

(b) Mean-Scale Hyperprior [Minnen et al., 2018].

(c) Channel-wise Autoregressive [Minnen and Singh, 2020].

Figure B.3: The curve-length ratio η v.s. the straight-path-length for randomly chosen image pairs, in three different nonlinear transform coding architectures [Ballé et al., 2018a, Minnen et al., 2018, Minnen and Singh, 2020]. In all cases, the curve lengths are close to the lengths of straight paths.

straight-path lengths. In all cases, the curve lengths are close to the straight-path lengths (η concentrated near 1), and this property appears to hold globally across randomly chosen image pairs.

Visualizing the filters of synthesis transforms

In Figure B.4, we visualize the learned filters of the base Mean-Scale Hyperprior architecture (panel a), and the proposed architectures with two-layer synthesis (panel b) and JPEG-like synthesis (panel c) (both paired with ELIC's analysis transform). We also visualize the top 20 PCA components learned on 10000 random 16×16 color image patches from COCO (panel d).

For the neural compression methods, we visualize the synthesis filters corresponding to the 20 latent channels with the highest bit-rates on average (as determined on a small batch of validation images); following [Duan et al., 2022], we produce the visualization for channel i as follows: let $\mathbf{e}$ be a 'basis' tensor of shape $[1, 1, C]$ (unit width/height) consisting of zeros except the ith channel, which equals 1; let $\mathbf{0}$ be a tensor of zeros with the same shape as $\mathbf{e}$; then the impulse response associated with channel i is computed as $g(\delta\mathbf{e}) - g(\mathbf{0})$, where δ is

(a) Top 20 filters of the Mean-Scale Hyperprior model, $\lambda = 0.00125$, giving bpp=0.10, psnr=27.3 on Kodak.

(b) Top 20 filters of the two-layer synthesis model, $\lambda = 0.00125$, giving bpp=0.12, psnr=27.3 on Kodak.

(c) Top 20 filters of the JPEG-like synthesis model, $\lambda = 0.00125$, giving bpp=0.12, psnr=27.0 on Kodak.

(d) Principal components with the 20 largest eigenvalues, extracted from random 16x16 natural image patches.

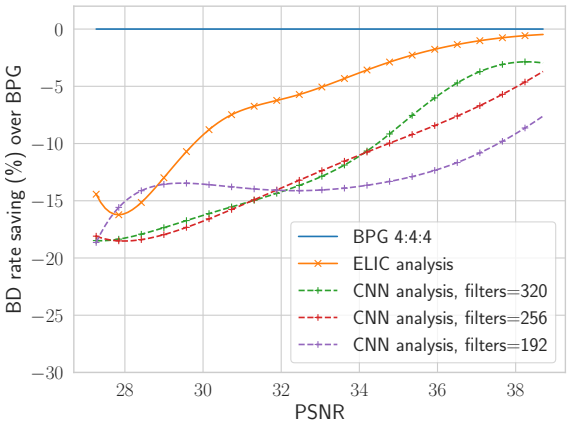
Figure B.4: Visualization of learned filters in various neural compression methods with varying synthesis transform complexity. The "filters" obtained from PCA are included for reference and show certain qualitative similarities. The filters are ordered by decreasing average bit-rate (a,b,c) or eigenvalue (d). See text description for more details.

a scaling factor which affects the color intensity when visualized. This results in a 16×16 colored image patch. We manually set a different δ for each architecture to result in a roughly comparable range of displayed colors, with $\delta \in [8, 20]$. We also apply a scaling factor when visualizing the principal components from PCA.

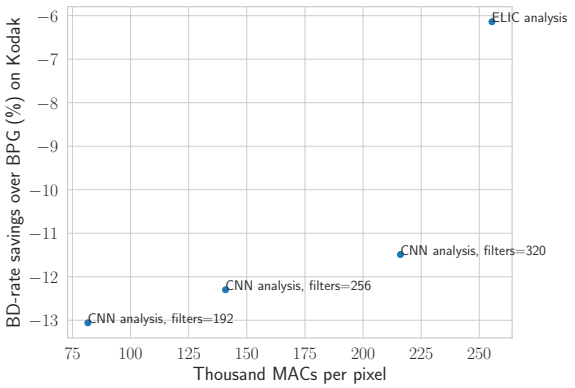
Ablation results.

The analysis transform. Here, we examine how different choices of the analysis transform affect the performance of our method based on the two-layer synthesis transform and ELIC's analysis transform.

We adopt the simpler CNN analysis transform from Mean-Scale Hyperprior [Minnen et al., 2018], which consists of 4 layers of convolutions with $F = 192$ filters each, except for the last layer which outputs $C = 320$ channels for the latent tensor. Fig. B.5 shows the resulting performance with varying F, in both BD-rate savings as well as computational complexity.



(a) BD-rate savings over BPG on Kodak, for various choices of analysis transforms.



(b) Aggregate BD-rate savings on Kodak, v.s. analysis transform complexity, measured in KMACs per pixel.

Figure B.5: Ablation results on the choice of analysis transform in the proposed two-layer synthesis architecture. Using the CNN analysis from Mean-Scale Hyperprior [Minnen et al., 2018] gave relatively worse R-D performance than the analysis transform from ELIC [He et al., 2022].

We see that the CNN analysis gave worse performance than ELIC analysis, and the gap can be closed to some extent by increasing F, but with diminishing returns and increasingly high encoding complexity.

Additional investigations. We present results giving additional insight into our method and how it compares to alternatives, evaluated on Kodak. The additional results are highlighted with blue legend titles in Figure B.6. First, we consider also applying SGA to the Hyperprior baseline; as shown by the solid blue line in Figure B.6, this also results in a sizable boost in R-D, even larger than what we observe for our more shallow decoders. We hypothesize that this may be caused by a relatively larger inference gap in the Hyperprior architecture than ours with shallow decoders.

Next, we show that simply scaling down existing neural compression models tends to result in worse performance than our approach. We consider two existing architectures: the mean-scale Hyperprior [Minnen et al., 2018] and ELIC [He et al., 2022], and slim down their synthesis transforms to match (to our best ability) the FLOPs of our two-layer shallow synthesis. For

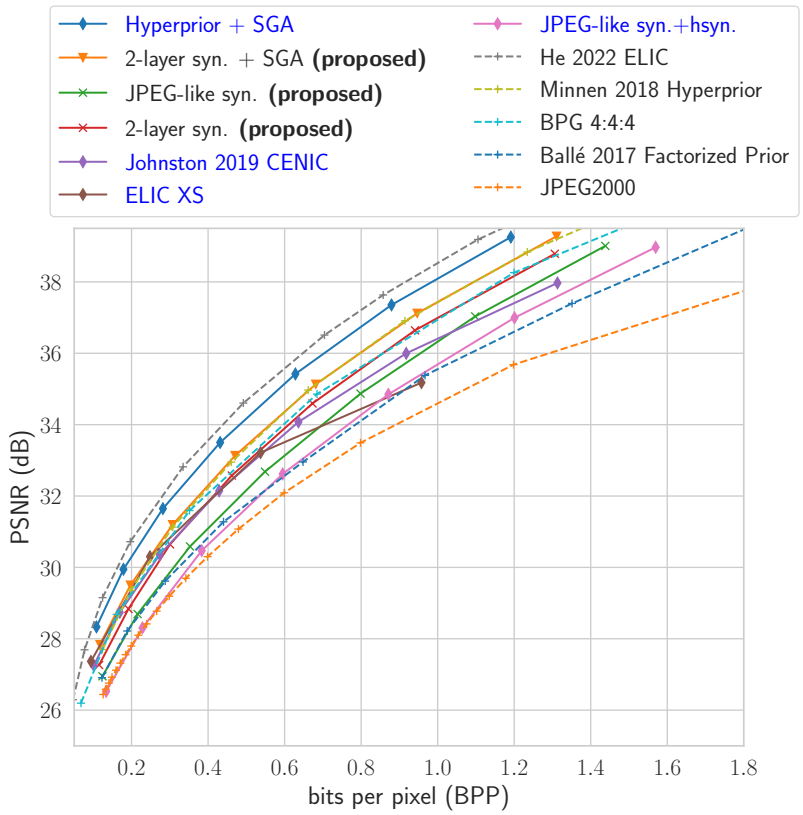


Figure B.6: Miscellaneous additional results, with blue legend labels to be distinguished from results in the main paper. We (1) apply SGA also to the Hyperprior baseline; (2) simply scale down existing NTC architectures to roughly match the FLOPs of our two-layer synthesis; (3) explore a JPEG-like hyper synthesis transform to improve its computational efficiency.

Hyperprior, we adopt a pruned synthesis transform given by CENIC [Johnston et al., 2019] (specifically, we use their architecture # 178, which uses about 7.3 KMACs/pixel, or about 1.4 times of our two-layer synthesis; we keep the hyper synthesis intact). For ELIC, we simply reduce the number of conv channels in the synthesis to be 32, so that it uses about 16.5 KMACs/pixel (we also keep its hyper synthesis intact). We train the resulting architectures from scratch; as shown by the purple ("Johnston 2019 CENIC") and brown curves ("ELIC XS"), this results in progressively worse R-D performance in the higher-rate regime compared to our two-layer synthesis (red curve).

Finally, we conduct a preliminary exploration of a JPEG-like architecture for the hyper synthesis transform. We implement this with a single transposed-conv layer with stride 4

and (6, 6) kernels. We applied it on top of our linear JPEG-like synthesis, and observe a 10% worse BD-rate (green curve → pink curve in Figure B.6) but nearly a 10-fold reduction in the hyper synthesis FLOPs (15.18 → 1.8 KMACs/pixel).

Additional R-D results

Below we include aggregate R-D results on the 100 test images from Tecnick [Asuni and Giachetti, 2014] and 41 images from the professional validation set of CLIC 2018 [5]. We additionally evaluate on the perceptual distortion LPIPS [Zhang et al., 2018]. Overall, we observe that our proposed two-layer synthesis with iterative encoding matches the Hyperprior performance when evaluated on PSNR, but under-performs by $8\% \sim 12\%$ (in BD-rate) when evaluated on perceptual metrics such as MS-SSIM or LPIPS. This is consistent with results on Kodak in Sec. 5.4.2.

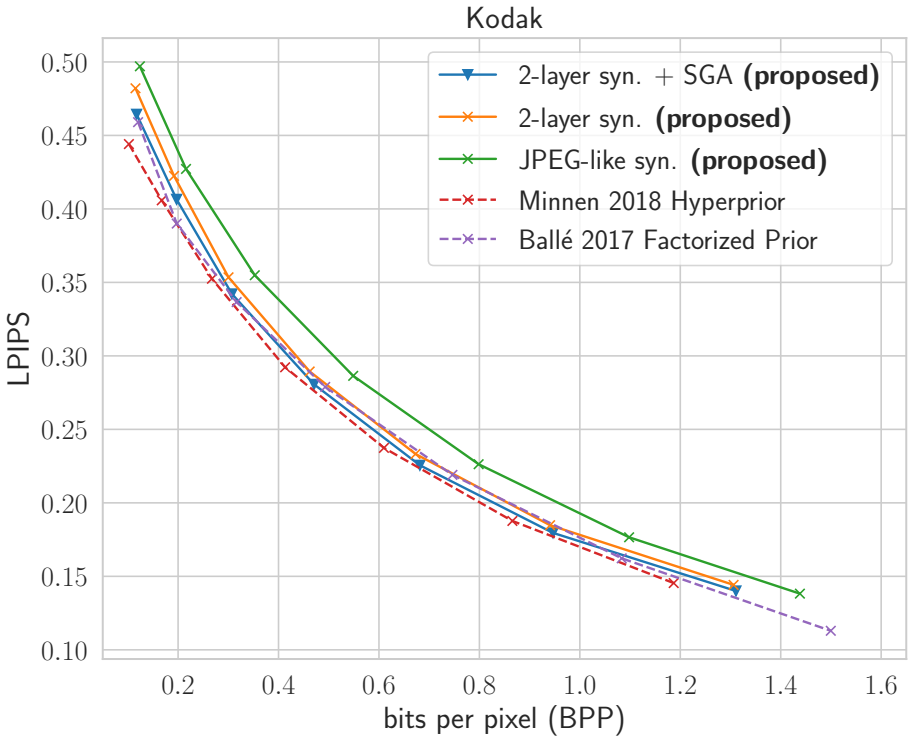


Figure B.7: Aggregate LPIPS v.s. BPP performance on Kodak.

[5] http://clic.compression.cc/2018/challenge/

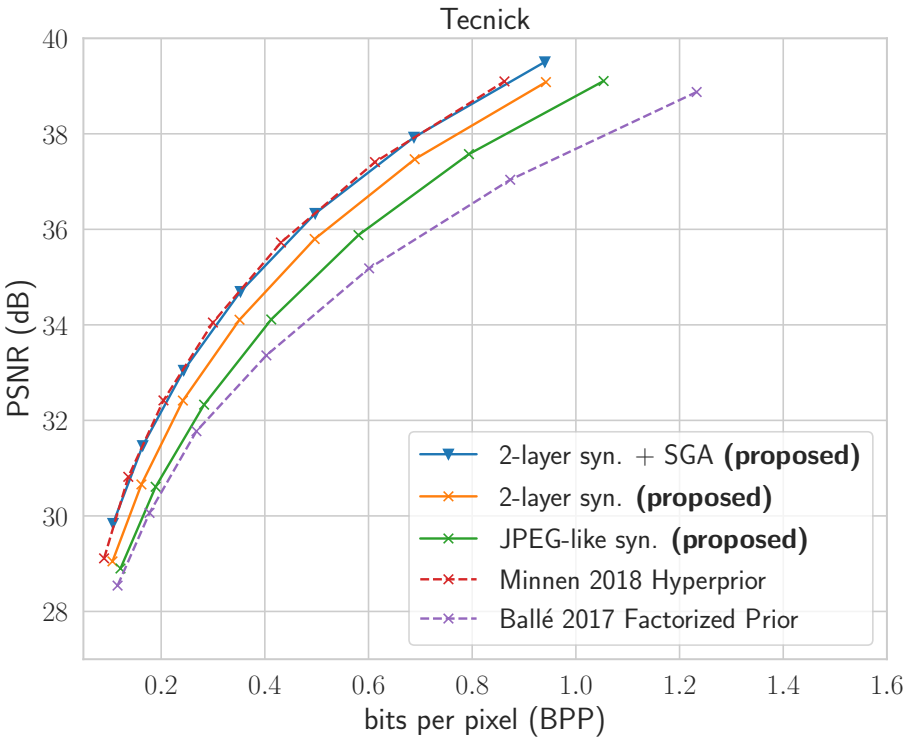


Figure B.8: Aggregate PSNR v.s. BPP performance on Tecnick.

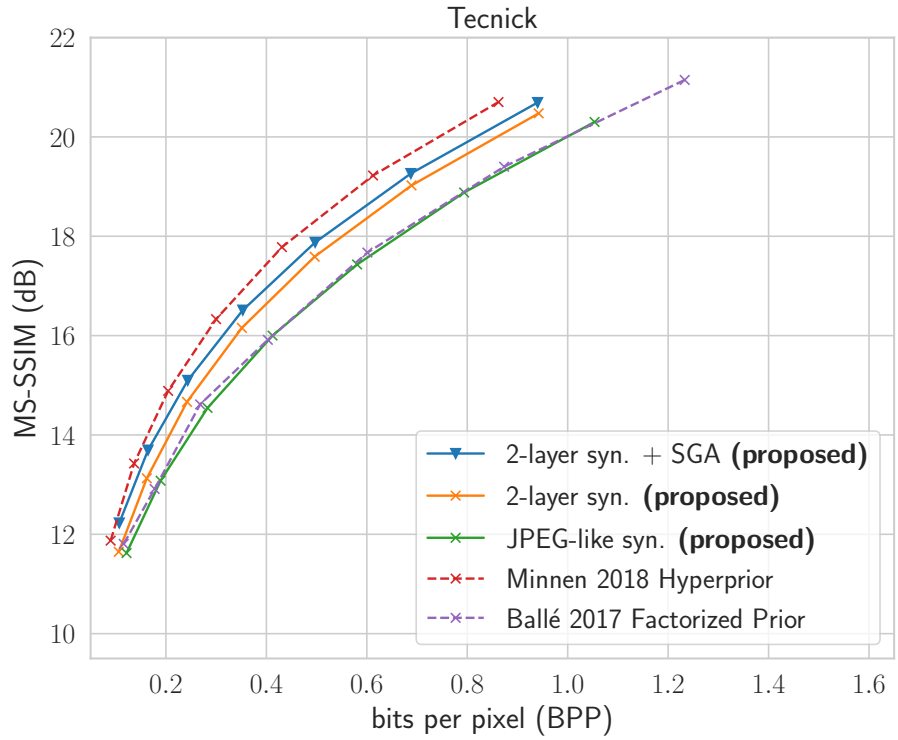


Figure B.9: Aggregate MS-SSIM v.s. BPP performance on Tecnick.

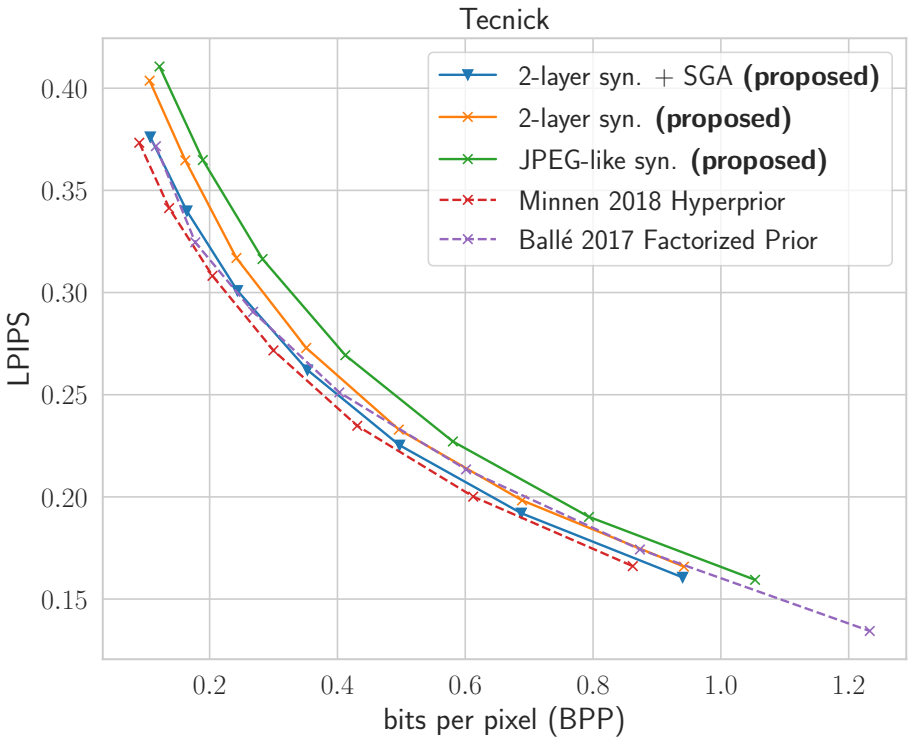


Figure B.10: Aggregate LPIPS v.s. BPP performance on Tecnick.

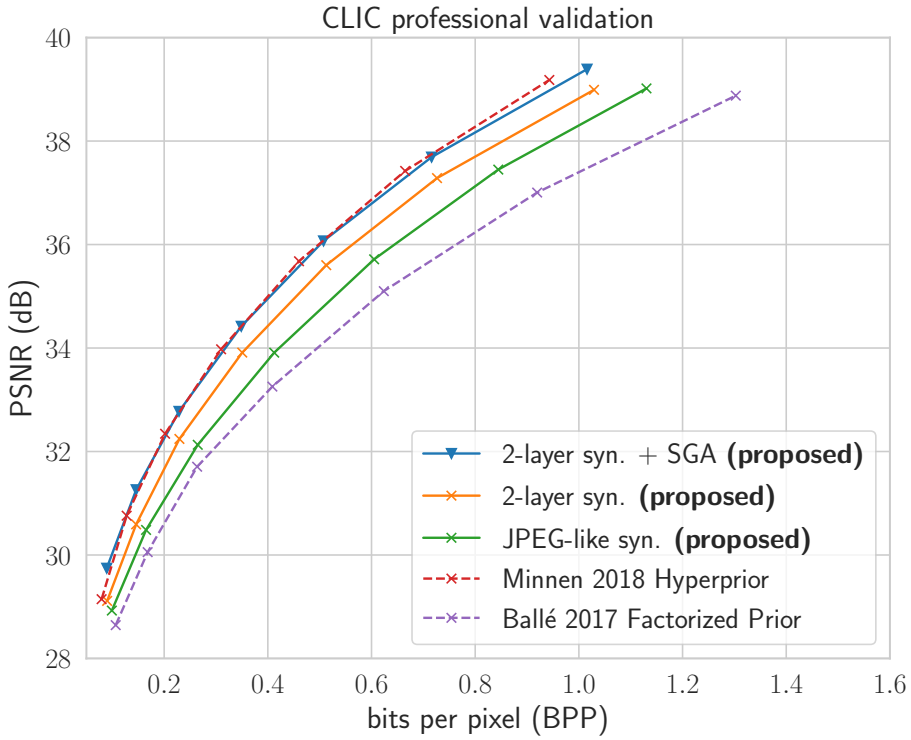


Figure B.11: Aggregate PSNR v.s. BPP performance on CLIC professional validation set.

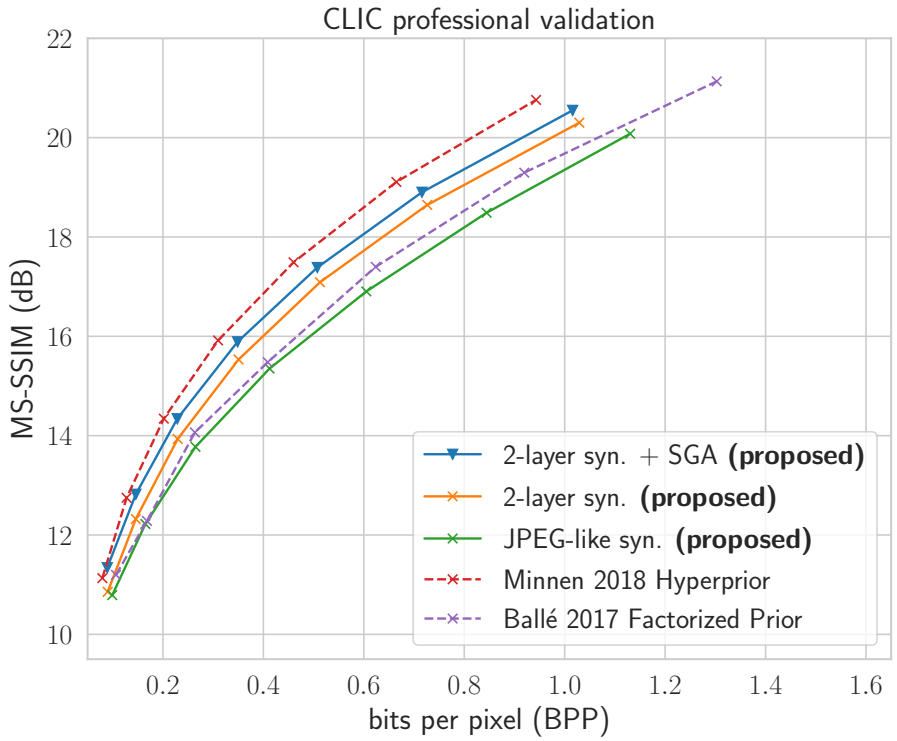


Figure B.12: Aggregate MS-SSIM v.s. BPP performance on CLIC professional validation set.

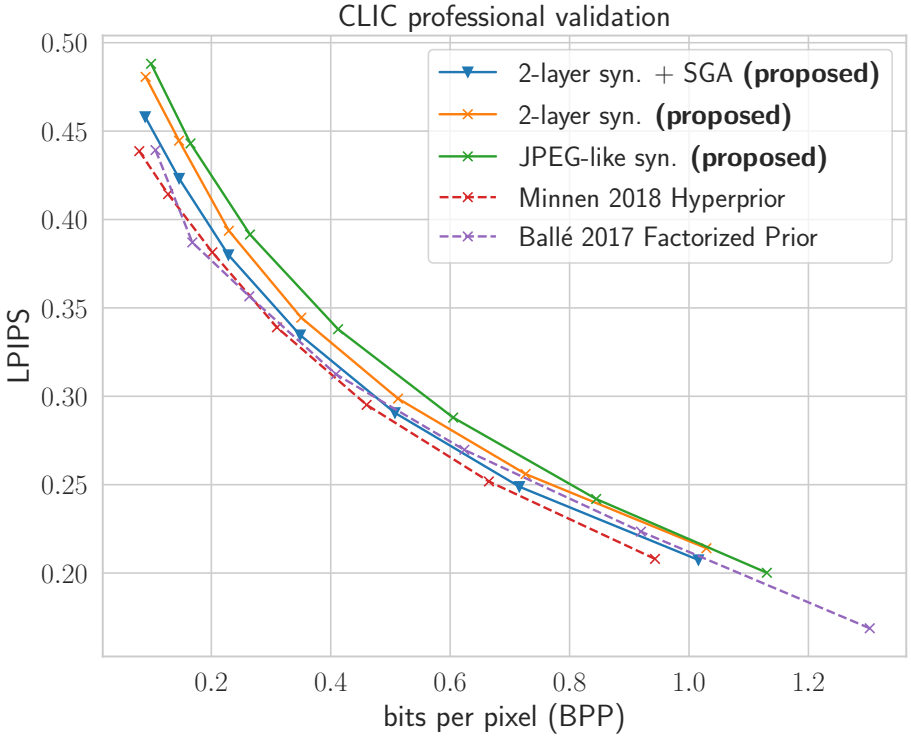


Figure B.13: Aggregate LPIPS v.s. BPP performance on CLIC professional validation set.

Appendix C

For Chapter 6, "Neural Network-Based Sandwich Bounds on the Rate-Distortion Function"

C.1 Technical Definitions and Prerequisites

In this work we consider the source to be represented by a random variable $X : \Omega \to \mathcal{X}$, i.e., a measurable function on an underlying probability space $(\Omega, \mathcal{F}, \mathbb{P})$, and P_X is the image measure of $\mathbb{P}$ under X. We suppose the source and reproduction spaces are standard Borel spaces, $(\mathcal{X}, \mathcal{A}_\mathcal{X})$ and $(\mathcal{Y}, \mathcal{A}_\mathcal{Y})$, equipped with sigma-algebras $\mathcal{A}_\mathcal{X}$ and $\mathcal{A}_\mathcal{Y}$, respectively. Below we use the definitions of standard quantities from Polyanskiy and Wu [2022].

Conditional distribution The notation $Q_{Y|X}$ denotes an arbitrary conditional distribution (also known as a Markov kernel), i.e., it satisfies

1. For any $x \in \mathcal{X}$, $Q_{Y|X=x}(\cdot)$ is a probability measure on $\mathcal{Y}$;

2. For any measurable set $B \in \mathcal{A}_{\mathcal{Y}}$, $x \to Q_{Y|X=x}(B)$ is a measurable function on $\mathcal{X}$.

Induced joint and marginal measures Given a source distribution P_X, each test channel distribution $Q_{Y|X}$ defines a joint distribution $P_X Q_{Y|X}$ on the product space $\mathcal{X} \times \mathcal{Y}$ (equipped with the usual product sigma algebra, $\mathcal{A}_{\mathcal{X}} \times \mathcal{A}_{\mathcal{Y}}$) as follows:

$$P_X Q_{Y|X}(E) := \int_{\mathcal{X}} P_X(dx) \int_{\mathcal{Y}} \mathbf{1}\{(x,y) \in E\} Q_{Y|X=x}(dy),$$

for all measurable sets $E \in \mathcal{A}_{\mathcal{X}} \times \mathcal{A}_{\mathcal{Y}}$. The induced y-marginal distribution P_Y is then defined by

$$P_Y(B) = \int_{\mathcal{X}} Q_{Y|X=x}(B) P_X(dx),$$

for all measurable sets $\forall B \in \mathcal{A}_{\mathcal{Y}}$.

KL Divergence We use the general definition of Kullback-Leibler (KL) divergence between two probability measures P, Q defined on a common measurable space:

$$KL(P\|Q) := \begin{cases} \int \log \frac{dP}{dQ} dP, & \text{if } P \ll Q \\ \infty, & \text{otherwise.} \end{cases}$$

$P \ll Q$ denotes that P is absolutely continuous w.r.t. Q (i.e., for all measurable sets E, $Q(E) = 0 \implies P(E) = 0$). $\frac{dP}{dQ}$ denotes the Radon-Nikodym derivative of P w.r.t. Q; for discrete distributions, we can simply take it to be the ratio of probability mass functions; and for continuous distributions, we can simply take it to be the ratio of probability density functions.

Mutual Information Given P_X and $Q_{Y|X}$, the mutual information $I(X;Y)$ is defined as

$$I(X;Y) := KL(P_X Q_{Y|X} \| P_X \otimes P_Y) = \mathbb{E}_{x\sim P_X}[KL(Q_{Y|X=x} \| P_Y)],$$

where P_Y is the Y-marginal of the joint $P_X Q_{Y|X}$, $P_X \otimes P_Y$ denotes the usual product measure, and $KL(\cdot\|\cdot)$ is the KL divergence.

For the mutual information upper bound, it's easy to show that

$$I_U(Q_{Y|X}, Q_Y) := \mathbb{E}_{x\sim P_X}[KL(Q_{Y|X=x} \| Q_Y)] = I(X;Y) + KL(P_Y \| Q_Y), \tag{C.1}$$

so the bound is tight when $P_Y = Q_Y$.

Obtaining $R(D)$ through the Lagrangian. For each $\lambda \geq 0$, we define the Lagrangian by incorporating the distortion constraint in the definition of $R(D)$ through a linear penalty:

$$\mathcal{L}(Q_{Y|X}, \lambda) := I(X;Y) + \lambda \mathbb{E}_{P_X Q_{Y|X}}[\rho(X,Y)], \tag{C.2}$$

and define its infimum w.r.t. $Q_{Y|X}$ by the function

$$F(\lambda) := \inf_{Q_{Y|X}} I(X;Y) + \lambda \mathbb{E}[\rho(X,Y)]. \tag{C.3}$$

Geometrically, $F(\lambda)$ is the maximum of the R-axis intercepts of straight lines of slope $-\lambda$, such that they have no point above the $R(D)$ curve [Csiszár, 1974c].

Define $D_{min} := \inf\{D' : R(D') < \infty\}$. Since $R(D)$ is convex, for each $D > D_{min}$, there exists a $\lambda \geq 0$ such that the line of slope $-\lambda$ through $(D, R(D))$ is tangent to the $R(D)$ curve, i.e.,

$$R(D') + \lambda D' \geq R(D) + \lambda(D) = F(\lambda), \quad \forall D'.$$

When this occurs, we say that λ *is associated to* D.

Consequently, the $R(D)$ curve is then the envelope of lines with slope $-\lambda$ and R-axis intercept $F(\lambda)$. Formally, this can be stated as:

Lemma C.1.1. (Csiszár [1974c, Lemma 1.2]; Gray [2011, Lemma 9.7]) *For every distortion tolerance* $D > D_{min}$*, where* $D_{min} := \inf\{D' : R(D') < \infty\}$*, it holds that*

$$R(D) = \max_{\lambda \geq 0} F(\lambda) - \lambda D \tag{C.4}$$

We can draw the following conclusions:

1. *For each* $D > D_{min}$*, the maximum above is attained iff* λ *is associated to* D*.*
2. *For a fixed* λ*, if* $Q^*_{Y|X}$ *achieves the minimum of* $\mathcal{L}(\cdot, \lambda)$*, then* λ *is associated to the point* $(\rho(Q^*_{Y|X}), I(Q^*_{Y|X}),)$*; i.e., there exists a line with slope* $-\lambda$ *that is tangent to the* $R(D)$ *curve at* $(\rho(Q^*_{Y|X}), I(Q^*_{Y|X}))$*, where we defined the shorthand* $\rho(Q_{Y|X}) := \mathbb{E}[\rho(X, Y)]$ *and* $I(Q_{Y|X}) := I(X; Y)$ *as induced by* $P_X Q_{Y|X}$*.*

C.2 Full Version of Theorem 6.4.1

We restate the theorem of Csiszár [1974c, Theorem 2.3] here for completeness.

Theorem C.2.1. (Csiszár [1974c, Theorem 2.3]; also see Kostina [2016, Theorem 1]) *Suppose that the following basic assumptions are satisfied.*

1. $R(D)$ *is finite for some* D*, i.e.,* $D_{min} := \inf\{D : R(D) < \infty\} < \infty$*;*
2. *The distortion metric* ρ *is such that there exists a finite set* $E \subset \mathcal{Y}$ *such that*

$$\mathbb{E}[\min_{y \in E} \rho(X, y)] < \infty$$

Then, for each $D > D_{min}$, it holds that

$$R(D) = \max_{g(x),\lambda} \{\mathbb{E}[-\log g(X)] - \lambda D\} \tag{C.5}$$

where the maximization is over $g(x) \geq 0$ and $\lambda \geq 0$ satisfying the constraint

$$\mathbb{E}\left[\frac{\exp(-\lambda\rho(X,y))}{g(X)}\right] = \int \frac{\exp(-\lambda\rho(x,y))}{g(x)} dP_X(x) \leq 1, \forall y \in \mathcal{Y} \tag{C.6}$$

Note: the basic assumption 2 is trivially satisfied when the distortion ρ is bounded from above; the maximization over $g(x) \geq 0$ can be restricted to only $1 \geq g(x) \geq 0$. Unless stated otherwise, we use log base e in this work, so the $R(D)$ above is in terms of nats (per sample).

Theorem C.2.1 can be seen as a consequence of Theorem 6.4.1 in conjunction with Lemma C.1.1. We implemented an early version of our lower bound algorithm based on Theorem C.2.1, generating each R-D lower bound by fixing a target D value and optimizing over both λ and g as in Equation C.5. However, the algorithm often diverged due to drastically changing λ. We therefore based our current algorithm on Theorem 6.4.1, producing R-D lower bounds by fixing λ and only optimizing over g (or u, in our formulation).

C.3 Theoretical Results

Theorem C.3.1. (A decoder-induced R-D function bounds the source R-D function from above.) *Let $X \sim P_X$ be a memoryless source subject to lossy compression under distortion ρ. Let $\mathcal{Z}$ be any measurable space ("latent space" in a VAE), and $\omega : \mathcal{Z} \to \mathcal{Y}$ any measurable function ("decoder" in a VAE). This induces a new lossy compression problem with $\mathcal{Z}$ being the reproduction alphabet, under a new distortion function $\rho_\omega : \mathcal{X} \times \mathcal{Z} \to [0, \infty), \rho_\omega(x, z) =$*

$\rho(x, \omega(z))$. *Define the corresponding ω-dependent rate-distortion function*

$$R_\omega(D) := \inf_{Q_{Z|X}:\mathbb{E}[\rho_\omega(X,Z)]\le D} I(X;Z) = \inf_{Q_{Z|X}:\mathbb{E}[\rho(X,\omega(Z))]\le D} I(X;Z).$$

Then for any $D \ge 0$, $R_\omega(D) \ge R(D)$. Moreover, the inequality is tight if ω is bijective (so that there is "no information loss").

Proof. Fix D. Take any admissible conditional distribution $Q_{Z|X}$ that satisfies $\mathbb{E}[\rho(X, \omega(Z))] \le D$ in the definition of $R_\omega(D)$. Define a new kernel $Q_{Y|X}$ between $\mathcal{X}$ and $\mathcal{Y}$ by $Q_{Y|X=x} := Q_{Z|X=x} \circ \omega^{-1}, \forall x \in \mathcal{X}$, i.e., $Q_{Y|X=x}$ is the image measure of $Q_{Z|X=x}$ induced by ω. Applying data processing inequality to the Markov chain $X \xrightarrow{Q_{Z|X}} Z \xrightarrow{\omega} Y$, we have $I(X;Z) \ge I(X;Y)$.

Moreover, since $Q_{Y|X}$ is admissible in the definition of $R(D)$, i.e.,

$$\mathbb{E}_{P_X Q_{Y|X}}[\rho(X,Y)] = \mathbb{E}_{P_X Q_{Z|X}}[\rho(X,\omega(Z))] \le D$$

we therefore have

$$I(X;Z) \ge I(X;Y) \ge R(D) = \inf_{Q_{Y|X}:\mathbb{E}[\rho(X,Y)]\le D} I(X;Y).$$

Finally, since $I(X;Z) \ge R(D)$ holds for any admissible $Q_{Z|X}$, taking infimum over such $Q_{Z|X}$ gives $R_\omega(D) \ge R(D)$.

To prove $R_\omega(D) = R(D)$ if ω is bijective, it suffices to show that $R(D) \ge R_\omega(D)$. We use the same argument as before. Take any admissible $Q_{Y|X}$ in the definition of $R(D)$. We can then construct a $Q_{Z|X}$ by the process $X \xrightarrow{Q_{Y|X}} Y \xrightarrow{\omega^{-1}} Z$. Then by DPI we have $I(X;Y) \ge I(X;Z)$. Morever, $Q_{Z|X}$ is admissible: $\mathbb{E}_{P_X Q_{Z|X}}[\rho(X,\omega(Z))] = \mathbb{E}_{P_X Q_{Y|X}}[\rho(X,\omega(\omega^{-1}(Y))] = \mathbb{E}_{P_X Q_{Y|X}}[\rho(X,Y)] \le D$. So $I(X;Y) \ge I(X;Z) \ge R_\omega(D)$. Taking infimum over such $Q_{Y|X}$ concludes the proof.

□

Remark. Although $\omega^{-1}(\omega(\circ)) = \text{Identity}(\circ)$ for any injective ω, our construction of $Q_{Z|X}$ in the last argument requires the inverse of ω to exist, so that additionally $\omega(\omega^{-1}(Y)) = Y$. Several learned image compression methods have advocated for the use of sub-pixel convolution, i.e, convolution followed by (invertible) reshaping of the results, over upsampled convolution in the decoder, in order to produce better reconstructions [Theis et al., 2017a, Cheng et al., 2020]. This can be seen as making the decoder more bijective, therefore reducing the gap of $R_\omega(D)$ over $R(D)$, in light of our above theorem.

Corollary C.3.1.1. (A suitable β-VAE defines an upper bound on the source R-D function). *Let $X \sim P_X$ be a data source, $\rho : \mathcal{X} \times \mathcal{Y} \to [0, \infty)$ a distortion function, and $\mathcal{Z}$ be any latent space. Consider any β-VAE consisting of a prior distribution Q_Z on $\mathcal{Z}$, and an encoder which specifies a conditional distribution $Q_{Z|X=x}$ for each x. Suppose further that the likelihood (observation) model is chosen to have density $p(x|z) \propto \exp\{-\rho(x, \omega(z)\}$ for some decoder function $\omega : \mathcal{Z} \to \mathcal{Y}$ (a common example being an isotropic Gaussian $p(x|z)$ with mean $\omega(z)$ and fixed variance, specified by a squared error distortion).*

Then the two terms of the negative ELBO — specifically, the "KL term"

$$\mathcal{R} := \mathbb{E}_{x \sim P_X}[KL(Q_{Z|X=x} \| Q_Z)],$$

and the "log-likelihood term" up to a constant,

$$\mathcal{D} := \mathbb{E}_{P_X Q_{Z|X}}[\rho(X, \omega(Z))] = \mathbb{E}_{P_X Q_{Z|X}}[-\log p(X|Z)] + \text{const},$$

define a point $(\mathcal{D}, \mathcal{R})$ that lies above the source $R(D)$-curve; i.e., $\mathcal{R} \geq R(\mathcal{D})$.

Proof.

$$\mathcal{R} \geq I_{P_X Q_{Z|X}}(X;Z) \geq \inf_{Q'_{Z|X}: \mathbb{E}_{P_X Q'_{Z|X}}[\rho_\omega(X,Z)] \leq \mathcal{D}} I_{P_X Q'_{Z|X}}(X;Z) =: R_\omega(\mathcal{D}) \geq R(\mathcal{D})$$

The first inequality is the variational upper bound on mutual information (Eq. C.1), the second inequality follows from the definition of the ω-induced R-D function (and the fact that $\mathbb{E}_{P_X Q_{Z|X}}[\rho_\omega(X,Z)] =: \mathcal{D}$), and the last inequality is Theorem C.3.1.

□

Remark. As also remarked by Theis et al. [2017a], not all distortion functions lead to the β-VAE interpretation above. For this to work, the function $\exp\{-\rho(x, \omega(z)\}$ must be normalizable in x (w.r.t. a suitable base measure on $\mathcal{X}$) for every z, and the normalizing constant may not depend on z.

Theorem C.3.2. (Basic properties of the proposed estimator C_k of the sup-partition function.) *Let $X_1, X_2, ... \sim P_X$ be a sequence of i.i.d. random variables. Let $\psi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$ be a measurable function. For each k, define the random variable $C_k := \sup_y \frac{1}{k} \sum_i \psi(X_i, y)$. Then*

1. *C_k is an overestimator of the sup-partition function c, i.e.,*
   $\mathbb{E}[C_k] = \mathbb{E}_{X_1,...,X_k}[\sup_y \frac{1}{k} \sum_i \psi(X_i, y)] \geq \sup_y \mathbb{E}[\psi(X, y)] =: c;$
2. *The bias of C_k decreases with increasing k, i.e.,*
   $\mathbb{E}[C_1] \geq \mathbb{E}[C_2] \geq ... \geq \mathbb{E}[C_k] \geq \mathbb{E}[C_{k+1}] \geq ... \sup_y \mathbb{E}[\psi(X, y)] = c;$
3. *If $\psi(x, y)$ is bounded and continuous in y, and if $\mathcal{Y}$ is compact, then C_k is strongly consistent, i.e., C_k converges to c almost surely (as well as in probability, i.e., $\lim_{k\to\infty} \mathbb{P}(|C_k - c| > \epsilon) = 0, \forall \epsilon > 0$), and $\lim_{k\to\infty} \mathbb{E}[C_k] = c$.*

Proof. We prove each in turn:

1. $\mathbb{E}[C_k] = \mathbb{E}[\sup_y \frac{1}{k}\sum_i \psi(X_i, y)] \geq \sup_y \mathbb{E}[\frac{1}{k}\sum_i \psi(X_i, y)] = \sup_y \mathbb{E}[\psi(X, y)] = c$

2. First, note that $\mathbb{E}[C_1] \geq \mathbb{E}[C_k]$ since

$$\mathbb{E}[C_1] = \mathbb{E}[\sup_y \psi(X_1, y)] = \mathbb{E}[\frac{1}{k}\sum_i \sup_y \psi(X_i, y)] \geq \mathbb{E}[\sup_y \frac{1}{k}\sum_i \psi(X_i, y)] = \mathbb{E}[C_k]$$

   We therefore have

$$\begin{aligned}
\mathbb{E}[C_{k+1}] &= \mathbb{E}[\sup_y \frac{1}{k+1}\sum_{i=1}^{k+1} \psi(X_i, y)] \\
&= \mathbb{E}[\sup_y \{\frac{1}{k+1}\sum_{i=1}^{k} \psi(X_i, y) + \frac{1}{k+1}\psi(X_{k+1}, y)\}] \\
&\leq \mathbb{E}[\sup_y \{\frac{1}{k+1}\sum_{i=1}^{k} \psi(X_i, y)\} + \sup_y \{\frac{1}{k+1}\psi(X_{k+1}, y)\}] \\
&= \frac{k}{k+1}\mathbb{E}[C_k] + \frac{1}{k+1}\mathbb{E}[C_1] \\
&\leq \mathbb{E}[C_k]
\end{aligned}$$

3. Define the shorthand $M_k(y) := \frac{1}{k}\sum_{i=1}^{k} \psi(X_i, y), \Psi(y) = \mathbb{E}[\psi(X, y)]$. Denote the sup norm on the set of continuous bounded functions on $\mathcal{Y}$ by $\|f(\cdot)\|_\infty := \sup_{y\in\mathcal{Y}} |f(y)|$. Then it holds that

$$\begin{aligned}
|\sup_y M_k(y) - \sup_y \Psi(y)| &= |\sup_y |M_k(y)| - \sup_y |\Psi(y)|| \\
&= |\|M_k(\cdot)\|_\infty - \|\Psi(\cdot)\|_\infty| \\
&\leq \|M_k(\cdot) - \Psi(\cdot)\|_\infty \\
&= \sup_y |M_k(y) - \Psi(y)|,
\end{aligned}$$

   where we made use of the fact that $\psi \geq 0$ in the first equality, and the reverse triangle inequality in the second-to-last step. By the uniform strong law of large numbers

[Ferguson, 2017],

$$\lim_{k\to\infty} \sup_y |M_k(y) - \Psi(y)| = 0$$

almost surely. Therefore, by the inequality we just showed, we also have

$$\lim_{k\to\infty} |\sup_y M_k(y) - \sup_y \Psi(y)| \leq \lim_{k\to\infty} \sup_y |M_k(y) - \Psi(y)| = 0$$

almost surely, i.e.,

$$\lim_{k\to\infty} C_k = \lim_{k\to\infty} \sup_y M_k(y) = \sup_y \mathbb{E}[\psi(X, y)] := c$$

almost surely. Then it trivially follows that C_k also converges to c in probability, and that $\lim_{k\to\infty} \mathbb{E}[C_k] = c$ by the bounded convergence theorem.

□

C.4 Detailed Lower Bound Method

Approximations for stochastic gradient ascent. Recall the lower bound objective obtained by replacing the sup-partition function c by the mean of the k-sample underestimator C_k:

$$\ell_k(\theta) := \mathbb{E}[-\log u_\theta(X)] - \log \mathbb{E}[C_k]$$

Unfortunately, we still cannot apply stochastic gradient ascent to ℓ_k, as two more difficulties remain, which we solve by further approximations. First, C_k is still hard to compute, as it is defined through a global maximization problem. We note that by restricting to a symmetric

ρ and $\mathcal{Y} = \mathcal{X}$, the maximization objective of C_k has the form of a reweighted kernel density estimate,

$$\frac{1}{k}\sum_{i=1}^{k}\psi(x_i, y) = \frac{1}{k}\sum_{i=1}^{k}\frac{1}{u(x_i)}\exp(-\lambda\rho(x_i, y)),$$

where each data point is reweighted by $\frac{1}{u(x)}$. For a squared error distortion, this becomes a Gaussian mixture density, $\frac{1}{k}\sum_i \psi(x_i, y) \propto \sum_i \pi_i \exp(-\lambda\|x_i - y\|^2)$, with centroids defined by the samples $x_1, ..., x_k$, and mixture weights $\pi_i = (ku(x_i))^{-1}$. The global mode of a Gaussian mixture can generally be found by hill-climbing from each of the k centroids, except in rare artificial examples [Carreira-Perpinan, 2000, 2020]; we therefore use this procedure to compute C_k, but note that other methods exist [Lee et al., 2019b, Carreira-Perpinan, 2007].

The second difficulty is that even if we could estimate $\mathbb{E}[C_k]$ (with samples of C_k computed by global optimization), the objective ℓ_k requires an estimate of its logarithm; a naive application of Jensen's inequality $-\log\mathbb{E}[C_k] \geq \mathbb{E}[-\log C_k]$ results in an *over*-estimator (as does the IWAE estimator), whereas we require a lower bound. Following Poole et al. [2019], we use the following basic lower bound of $-\log(x)$ by its linearization around some point α, i.e.,

$$-\log(x) \geq -x/\alpha - \log(\alpha) + 1, \forall x, \alpha > 0$$

with equality achieved by $\alpha = x$. This results in the final lower bound objective we optimize:

$$\tilde{\ell}_k(\theta) := \mathbb{E}[-\log u_\theta(X)] - \mathbb{E}[C_k]/\alpha - \log\alpha + 1. \tag{C.7}$$

Since the linear underestimator of $-\log\mathbb{E}[C_k]$ is tightest near $\alpha = \mathbb{E}[C_k]$, in our algorithm we set α adaptively to an estimate of $\mathbb{E}[C_k]$ (separately from the $\mathbb{E}[C_k]$ term we estimate in the objective function).

Algorithm 2: Example implementation of the proposed algorithm for estimating rate-distortion lower bound $R_L(D)$.

Requires: $\lambda > 0$, model u_θ (e.g., a neural network) parameterized by θ, batch sizes k, m, and gradient ascent step size ϵ.

while $\tilde{\ell}_k$ *not converged* **do**

`// Draw` $2m$ `samples of` C_k`; use the last` m `samples to set` α`, and the first` m `samples to estimate` $\mathbb{E}[C_k]$ `in` $\tilde{\ell}_k$`.`

for $j := 1$ **to** $2m$ **do**

Draw k data samples, $\{x_1^j, ..., x_k^j\}$

$y^j, C_k^j :=$ compute_$C_k(\theta, \lambda, \{x_1, ..., x_k\})$

end

$\alpha := \frac{1}{m} \sum_{j=m+1}^{2m} C_k^j$

`// Compute objective` $\tilde{\ell}_k(\theta)$ `and update` θ `by gradient ascent.`

with GradientTape() as tape:

$E := \frac{1}{m} \sum_{j=1}^{m} \gamma_k(y^j, \theta, \{x_1^j, ..., x_k^j\})$ `// Estimate of` $\mathbb{E}[C_k]$

$\tilde{\ell}_k := -\frac{1}{m} \sum_{j=1}^{m} \frac{1}{k} \sum_{i=1}^{k} \log u_\theta(x_i^j) - \frac{1}{\alpha} E - \log \alpha + 1$

gradient := tape.gradient($\tilde{\ell}_k$, θ)

$\theta := \theta + \epsilon$ gradient

end

Subroutine $\gamma_k(y, \theta, \lambda, \{x_1, ..., x_k\})$:

`// Evaluate the global maximization objective at` y`.`

Return $\frac{1}{k} \sum_{i=1}^{k} \exp\{-\lambda \rho(x_i, y)\} / u_\theta(x_i)$

Subroutine compute_$C_k(\theta, \lambda, \{x_1, ..., x_k\})$:

`// Compute the global optimum of` $\gamma_k(y)$`, assuming squared distortion` $\rho(x, y) \propto \|x - y\|^2$`.`

opt_loss := $-\infty$

for $i := 1$ **to** k **do**

`// Run gradient ascent from the` i`th mixture component.`

$y := x_i$

while y *not converged* **do**

with GradientTape() as tape:

loss := $\gamma_k(y, \theta, \lambda, \{x_1, ..., x_k\})$

gradient := tape.gradient(loss, y)

$y := y + \epsilon$ gradient

end

if *loss* > *opt_loss* **then**

opt_loss := loss

$y^* := y$

end

end

C_k = opt_loss

return (y^*, C_k)

The algorithm. We give a pseudo-code implementation of the algorithm in Algorithm 2. The code largely follows Python semantics, and assumes an auto-differentiation package is available, such as GradientTape as provided in `Tensorflow`. We use γ_k to denote the function in the supremem definition of C_k , i.e., $\gamma_k(y) := \frac{1}{k}\sum_{i=1}^{k} \exp\{-\lambda\rho(x_i, y)\}/u_\theta(x_i)$.

For a given $\lambda > 0$ and model $u_\theta : x \to \mathbb{R}^+$, the lower bound algorithm works by (stochastic) gradient ascent on the objective $\tilde{\ell}_k(\theta)$ (Eq. 6.8) w.r.t. the model parameters θ. To compute the gradient, we need an estimate of $\mathbb{E}[C_k]$ (ultimately $\log \mathbb{E}[C_k]$) as part of the objective, which requires us to draw m samples of C_k. In the pseudo-code, a separate set of m samples of C_k are also drawn to set the linear expansion point α. In our actual implementation, we set α to an exponential moving average of $\mathbb{E}[C_k]$ estimated from previous iterations (e.g., replacing line 7 of the pseudo-code by $\alpha := 0.2\alpha + 0.8E$), so only m samples of C_k need to be drawn for each iteration of the algorithm. We did not find this approximation to significantly affect the training or results.

Recall $C_k(\theta)$ is defined by the result of maximizing w.r.t. y. When computing the gradient with respect to θ, we differentiate through the maximization operation by evaluating $C_k = \gamma_k(y^*)$ on the forward pass, using the argmax y^* found by the subroutine compute_C_k. This is justified by appealing to a standard envelope theorem [Milgrom and Segal, 2002].

By Lemma C.1.1, each u^*_θ trained with a given value of λ yields a linear under-estimator of $R(D)$,

$$R^\lambda_L(D) = -\lambda D + \tilde{\ell}_k(\theta^*).$$

We obtain the final $R_L(D)$ lower bound by taking the upper envelope of all such lines, i.e.,

$$R_L(D) := \max_{\lambda \in \Lambda} R_L^\lambda(D),$$

where Λ is the set of λ values we trained with.

Tips and tricks:

To avoid numerical issues, we always parameterize $\log u$, and all the operations involving C_k and α are performed in the log domain; e.g., we optimize $\log \gamma_k$ instead of γ_k (which does not affect the result since log is monotonically increasing), and use logsumexp when taking the average of multiple C_k samples in the log domain.

During training, we only run an approximate version of the global optimization subroutine compute$_C_k$ for efficiency, as follows. Instead of hill-climbing from each of the k mixture component centroids, we only do so from the t highest-valued centroids under the objective γ_k, for a small number of t (say 10). This approximation is not used when reporting the final results.

C.5 Additional Experimental Details and Results

C.5.1 Computational Aspects

Our methods are implemented using the `Tensorflow` library. Our experiments with learned image compression methods additionally used the `tensorflow-compression`[1] library. Our experiments on images were run on Titan RTX GPUs, while the rest of the experiments were run on Intel(R) Xeon(R) CPUs. We used the Adam optimizer for gradient based

[1] https://github.com/tensorflow/compression

optimization in all our experiments, typically setting the learning rate between $1e-4$ and $5e-4$. Training the β-VAEs for the upper bounds required from a few thousand gradient steps on the lower-dimension problems (under an hour), to a few million gradient steps on the image compression problem (a couple of days; similar to reported in Minnen and Singh [2020]). With our approximate mode finding procedure (starting hill climbing only from a small number of centroids, see Sec. C.4), the lower bound models required less than 10000 gradient steps to converge in all our experiments.

C.5.2 Gaussians

Data Here the data source is an n-dimensional Gaussian distribution with a diagonal covariance matrix, whose parameters are generated randomly. We sample each dimension of the mean uniformly randomly from $[-0.5, 0.5]$ and variance from $[0, 2]$. The ground-truth R-D function of the source is computed analytically by the reverse water-filling algorithm [Cover and Thomas, 2006].

Upper Bound Experiments For the $\mathcal{Z} = \mathcal{Y}$ (no decoder) experiment, we chose Q_Y and $Q_{Y|X}$ to be factorized Gaussians; we let the mean and variance vectors of Q_Y be trainable parameters, and predict the mean and variance of $Q_{Y|X=x}$ by one fully connected layer with $2n$ output units, using softplus activation for the n variance components.

For our experiments involving a decoder, we parameterize the variational distributions Q_Z and $Q_{Z|X}$ similarly to before, and use an MLP decoder with one hidden layer (with the number of units equal the data dimension n, and leaky ReLU activation) to map from the $\mathcal{Z}$ to $\mathcal{Y}$ space. We observe the best performance with a linear (or identity) decoder and a simple linear encoder; using deeper networks with nonlinear activation required more training iterations for SGD to converge, and often to a poorer upper bound. In fact, for a factorized

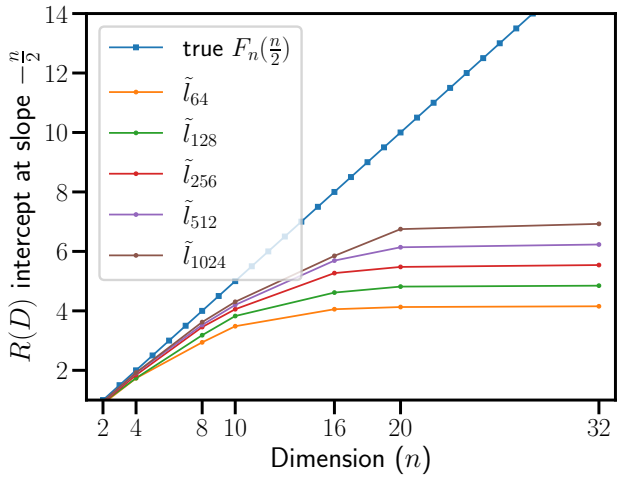


Figure C.1: R-axis intercept estimates at (negative) slope $\lambda = \frac{n}{2}$ from our lower bound algorithm, trained with increasing n and k. The R-axis intercept of an optimally tight linear lower bound, $F_n(\frac{n}{2})$, increases linearly as a function of n; however, we see that the best achievable R-axis intercept achieved with a particular setting of k, $\tilde{\ell}_k$, plateaus out at $\log_k$ as n increases.

Gaussian source, it can be shown analytically that the optional $Q^*_{Y|X=x}$ is a Gaussian whose mean depends linearly on x, and an identity (no) decoder is optimal.

Lower Bound Experiments In our lower bound experiments, we parameterize $\log u$ by an MLP with 2 hidden layers with $20n$ hidden units each and SeLU activation, where n is the dimension of the Gaussian source. We fixed $k = 1024$ for the results with varying n in Figure 6.2a-*bottom*. We vary both k and n in the additional experiment below.

To investigate the necessary k needed to achieve a tight lower bound, and its relation to λ or the Gaussian source dimension n, we ran a series of experiments with k ranging from 64 to 1024 on increasingly high dimensional standard Gaussian sources, each time setting $\lambda = \frac{n}{2}$. For an n-dimensional standard Gaussian, the true R-intercept, $F_n(\lambda)$, has an analytical form; in particular, $F_n(\frac{n}{2}) = \frac{n}{2}$. In Figure C.1, we plot the final objective estimate, $\tilde{\ell}_k$, using the converged MLP model, one for each k and n. As we can see, the maximum achievable $\tilde{\ell}_k$ plateaus to the value $\log k$ as we increase n, and for sufficiently high dimension (e.g., $n = 20$

here), doubling k only brings out a constant ($\log 2$) improvement in the objective. This phenomenon relates to the over-estimation bias of C_k when k is too low compared to λ or the "spread" of the data, and can be understood as follows.

For a given sample size k, there is a "catestrophic" regime produced by increasingly large λ (corresponding to an exceedingly narrow Gaussian kernel), or increasingly high (intrinsic) dimension of the data, so that the k data samples appear very far apart. Numerically, this is exhibited by very quick termination of the k hill-climbing runs when computing C_k (subroutine compute_C_k in Algorithm. 2), since the mixture components are well separated, and the mixture centroids are nearly stationary points of the mixture density. The value of the mixture density at each centroid x_i can be approximated as $\gamma_k(x_i) = \frac{1}{k}\frac{1}{u(x_i)} + \frac{1}{k}\sum_{j\neq i}\exp\{-\lambda\rho(x_j, x_i)\}/u_\theta(x_j) \approx \frac{1}{ku(x_i)}$, since $\exp\{-\lambda\rho(x_j, x_i)\} \approx 0$ for all $j \neq i$. The maximization problem defining C_k then essentially reduces to checking which mixture component is the highest, and returning the corresponding centroid (or a point very close to it), so that $C_k \approx \sup_{i=1,\ldots,k}\frac{1}{ku(x_i)}$. This implies

$$\mathbb{E}[C_k] \approx \mathbb{E}[\sup_{i=1,\ldots,k}\frac{1}{ku(X_i)}] \leq \mathbb{E}[\sup_x \frac{1}{ku(x)}] = \sup_x \frac{1}{ku(x)},$$

therefore

$$\ell_k := \mathbb{E}[-\log u(X)] - \log \mathbb{E}[C_k] \tag{C.8}$$

$$\leq \mathbb{E}[-\log u(X)] + \log k + \inf_x \log u(x) \tag{C.9}$$

$$= \log k + \mathbb{E}[-\log u(X) + \inf_{x'} \log u(x')] \leq \log k. \tag{C.10}$$

Thus, when these approximations hold, the maximum achievable lower bound objective $\tilde{\ell}_k$ cannot exceed $\log k$. On sources such as high-dimension (e.g., $n = 10000$) Gaussians, or 256×256 patches of natural images, λ needs to be on the order of 10^6 or higher to target even the low-distortion region of the R-D curve, and the above analysis well describes the

behavior of the lower bound algorithm for any value of k we feasibly experimented with.

C.5.3 Particle Physics

Data We borrowed the $Z \to e^+e^-$ dataset from Howard et al. [2021], containing 331699 data observations. Each $n = 16$-dimensional data vector corresponds to an independent Z-boson decay event, and is the concatenation of the four-momenta of an electron (e^-) and positron (e^+) both in the theoretical prior space and observed space. The 2D marginal distribution (where we additionally compare with the BA algorithm) is formed by taking the two data coordinates with the least absolute value of correlation. A histogram is given in Figure C.2a.

Unlike in the Gaussian experiments, where we generate new random samples from the source on the fly for training and evaluation, here the true underlying source is only approximated by a finite number of samples in the given dataset. Since our R-D upper/lower bounds have the form of expectations of learned functions w.r.t. the *true* underlying source (explained in more detail in Section. C.6), we create a separate held-out test set of 10000 samples for our final estimates of R-D bounds using the trained models. We also use a small subset of the training data as a validation set for model selection, typically using the most expressive model we can afford without overfitting. We use the same train/test split and the same model architectures on the 2D marginal data as on the $n = 16$ data.

Upper Bound Experiments For our R-D upper bound model, we use a VAE with a MLP decoder, a normalizing flow "prior" Q_Z, and an MLP encoder computing a factorized Gaussian $Q_{Z|X}$. We set the latent space to have the same dimensionality n as the data. The encoder and decoder MLPs have three layers with $[500, 500, l]$ units in each, where $l = n$ for the deocder MLP and $l = 2n$ for the encoder MLP (as it computes both a mean and a variance

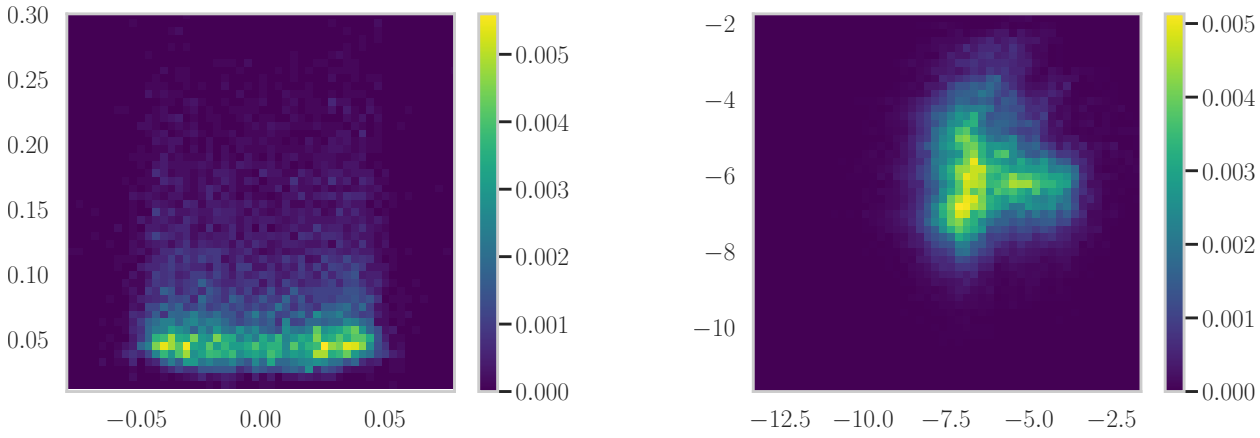
vector for $Q_{Z|X}$); we use softplus activation in each hidden layer. Q_Z is parameterized as a base Gaussian distribution transformed by three stacks of MAF [Papamakarios et al., 2017].

For the Nonlinear Transform Coding (NTC) [Ballé et al., 2021] model, we use a VAE with a similar 500-500-n MLP architecture for the encoder and decoder (thus the latent space has the same dimensionality n as the data), but modify the variational distributions to simulate quantization and end-to-end compression. As in [Ballé et al., 2018a, Ballé et al., 2021], we let $Q_{Z|X}$ be a factorized uniform distribution with unit width, and Q_Z be a factorized neural density model convolved with a uniform noise.

For the BA algorithm on the 2D marginal, we obtain discretized alphabets and source probabilities as follows. First, we determine the smallest box set S in $\mathbb{R}^2$ containing all the data samples (simply the Cartesian product of sample ranges along the x and y axes); next, we finely tile up S using small rectangle bins, and compute a histogram of data samples over them. We then let the source and reproduction alphabets to be the bin centers, and set the source distribution to be the frequency of samples in each bin. Finally, for computation efficiency, we remove the bin centers from the source alphabet associated with 0 samples; this does not affect the results.

We ran a maximum of 2000 steps of the BA algorithm on the test set, terminating early if the objective has not improved by more than 0.0001% in consecutive steps.

Lower Bound Experiments We parameterize our lower bound model $\log u$ by an MLP with 3 hidden layers with 200 hidden units each and SeLU activation. We fixed $k = 2048$.



(a) (b)

Figure C.2: Normalized histograms of the 2D marginals of the particle physics (C.2a) and speech (C.2b) datasets. The quantization bins and the associated empirical PMFs are also used define the discretized 2D source distributions on which the BA algorithm is run.

C.5.4 Speech

Data We use the Free Spoken Digit Dataset [Jackson et al., 2018] consisting of recordings of spoken digits from 'zero' to 'nine' at a sampling rate of 8kHz. We use a subset of the recordings with a randomly chosen speaker id ('theo'), and keep the original train-test split [2] consisting of 450 and 50 audio files for the train and test sets, respectively. From each audio file, we extracted Short-time Fourier Transform (STFT) coefficients using Tensorflow's `tf.signal.stft` routine with a 63-sample FFT window (`frame_length=63`) in 1-sample steps (`frame_step=1`), took the magnitude and converted to log domain, a common pre-processing step for computing spectrograms and Mel-frequency cepstral coefficients. The resulting $n = 33$-dimensional Fourier feature vectors are then shuffled across time steps and treated as i.i.d. samples in our experiments, with over 1 million samples in the training set and 120000 samples in the test set. We follow the same procedure as in the physics experiment, creating a 2D marginal from the two coordinates with the least absolute correlation; a histogram is given in Figure C.2b.

[2] https://github.com/Jakobovski/free-spoken-digit-dataset/#usage

Experiments We use the same model architectures and experimental procedures/hypermarameters as in the physics experiment.

C.5.5 Banana-shaped Source

Data The 2D banana-shaped source [Ballé et al., 2021] is a RealNVP transform of a 2D Gaussian. [3] See Figure 6.3 for a plot of its density function.

We embed the banana source in a higher dimension n by multiplying with a randomly sampled $n \times 2$ matrix. This is done by simply passing samples of the 2D banana source through a linear MLP layer with n output units and no activation, with its weight matrix randomly drawn from a Gaussian by the Glorot initialization procedure.

Upper Bound Experiments We reproduced the NTC experiment using the same compressive autoencoder as in Ballé et al. [2021], with a 2-dimensional latent space and two-layer MLPs for both the encoder and decoder, with 100 hidden units and softplus activation in each hidden layer. For our upper bound model, we use the same MLP architecture as in Ballé et al. [2021], but parameterize Q_Z by a MAF [Papamakarios et al., 2017] and $Q_{Z|X}$ by a factorized Gaussian (doubling the number of output of the encoder), similar to on previous experiments.

For the higher-dimension embeddings of the banana-shaped source, we use the same β-VAE as on the 2D source, but set the number of hidden units in each hidden layer to $100n$, where n is the dimension of the source. We cap the number of hidden units per layer at 2000 for $n > 20$, and do not find this to adversely affect the results.

[3] Source code taken from the authors' GitHub repo https://github.com/tensorflow/compression/blob/66228f0faf9f500ffba9a99d5f3ad97689595ef8/models/toy_sources/toy_sources.ipynb.

Lower Bound Experiments We use an MLP with three hidden layers and SeLU activations for the $\log u$ model; as in the upper bound models, here we set the number of hidden units in each layer to be $100n$, and cap it at 1000. In all the experiments we set $k = 1024$.

As illustrated by Figure 6.5, the tightness of our sandwich bounds do not appear to be affected by the dimension n of the data (we verified this up to $n = 1000$), unlike on the Gaussian experiment where for a fixed k the lower bound became increasingly loose with higher n.

C.5.6 GAN-Generated Images

Data We adopt the same setup as in the GAN experiment by Pope et al. [2021]. We use a BigGAN [Brock et al., 2019] pretrained on ImageNet at 128×128 resolution (so that $n = 128 \times 128 \times 3 = 49152$), downloaded from `https://tfhub.dev/deepmind/biggan-deep-128/1`. To generate an image of an ImageNet category, we provide the corresponding one-hot class vector as well as a noise vector to the GAN. Following Poole et al. [2019], we control the intrinsic dimension d of the generated images by setting all except the first d dimensions of the 128-dimension noise vector to zero, and use a truncation level of 0.8 for the sample diversity factor. Since the GAN generates values in $[-1, 1]$, we rescale them linearly to $[0, 1]$ to correspond to images, so that the alphabets $\mathcal{X} = \mathcal{Y} = [0, 1]^n$. As in [Pope et al., 2021], we experimented with images from the ImageNet class `basenji`. In Figure C.3 and C.4, we plot additional random samples, for $d = 2$ and $d = 4$.

Upper Bound Experiments We implemented a version of ResNet-VAE following the appendix of Kingma et al. [2016], using the 3×3 convolutions of Cheng et al. [2020] in the encoder and decoder of our model. Our model consists of 6 layers of latent variables, and implements bidirectional inference using factorized Gaussian variational distributions at each stochastic layer. During an inference pass, an input image goes through 6 stages of

Figure C.3: Random samples of `basenji` from a BigGAN, $d = 2$.

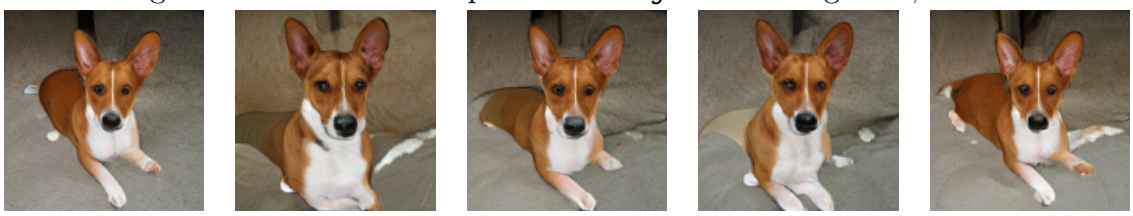


Figure C.4: Random samples of `basenji` from a BigGAN, $d = 4$.

convolution followed by downsampling (each time reducing the height and width by 2), and results in a stochastic tensor at each stage. In encoding order, the stochastic tensors have decreasing spatial extent and an increasing number of channels equal to 4, 8, 16, 32, 64, and 128. The latent tensor Z_0 at the topmost generative hierarchy is flattened and modeled by a MAF prior Q_{Z_0} (aided by the fact that all the images have a fixed shape).

Lower Bound Experiments We parameterize the $\log u$ model by a simple feedforward convolutional neural network. It contains three convolutional layers with 4, 8, and 16 filters, each time downsampling by 2, followed by a fully connected layer with 1024 hidden units. Again we use SeLU activation in the hidden units, and set $k = 2048$ in all the experiments.

C.5.7 Natural Images

Data For signals such as images or video, which can have arbitrarily high spatial dimension, it is not immediately clear how to define i.i.d. samples. But since our focus is on image compression, we follow the common practice of training convolutional autoencoders on random 256×256-pixel crops of high resolution images as in learned image compression research [Ballé

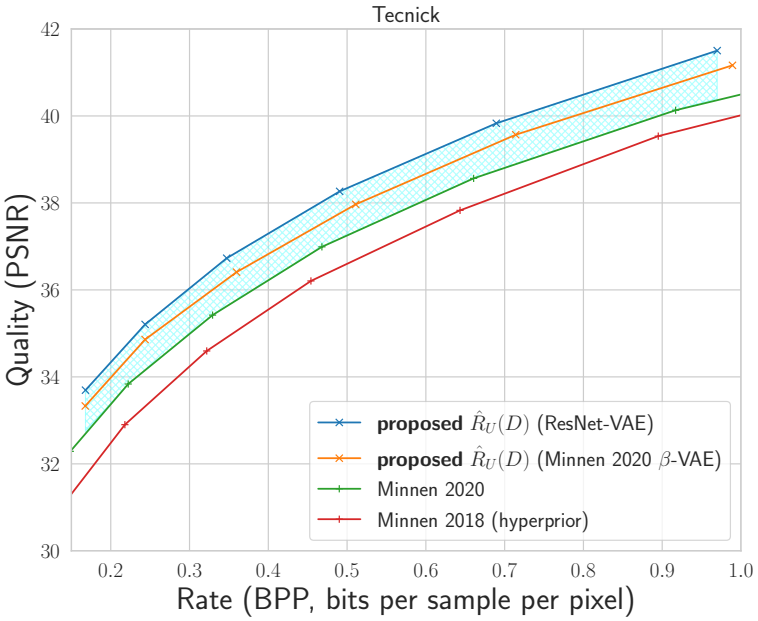


Figure C.5: Quality-rate curves on the Tecnick dataset.

et al., 2017], noting that current methods cannot effectively exploit spatial redundancies larger than this scale [Minnen and Singh, 2020]. We can then use the trained models to estimate R-D upper bounds on the image-generating distributions underlying standard benchmark datasets such as Kodak Kodak [1993] and Tecnick Asuni and Giachetti [2014].

As a representative dataset of natural images, we used images from the COCO 2017 [Lin et al., 2014] training set that are larger than 512×512, and downsampled each by a random factor in $[0.6, 1]$ to remove potential compression artifacts from JPEG. We trained image compression models from [Minnen et al., 2018, Minnen and Singh, 2020] on our dataset and were able to closely reproduce their performance.

Upper Bound Experiments Instead of working with variational distributions over the set of pixel values $\mathcal{Y} = \mathcal{X} = \{0, 1, ..., 255\}^n$, which would require specialized tools for representing and optimizing high-dimensional discrete distributions [Salimans et al., 2017, Hoogeboom et al., 2019, Maddison et al., 2016, Jang et al., 2016], here we parameterize the variational distributions in Euclidean latent spaces for simplicity (see Sec. 6.3).

We applied our upper bound algorithm based on two VAE architectures:

1. The learned image compression model from Minnen and Singh [2020] [4]. This corresponds to a hierarchical VAE with two layers of latent variables, with the lower level latents modeled by a channel-wise autoregressive prior conditioned on the top-level latents. We adapt it for our purpose, by using factorized Gaussians for the variational posterior distributions $Q_{Z|X}$ (instead of factorized uniform distributions used to simulate rounding in the original model), and no longer convolving the variational prior Q_Z with a uniform distribution as in the original model.

2. The ResNet-VAE from our experiment on GAN images. Following learned image compression models, we maintain the convolutional structure of the topmost latent tensor (rather than flattening it in the GAN experiments) in order to make the model applicable to images of variable sizes. Instead of a MAF, we model the topmost latent tensor by a deep factorized (hyper-)prior Q_{Z_0} as proposed by Ballé et al. [2018a].

For the β-VAE based on Minnen and Singh [2020], $\dim(\mathcal{Z}) \approx 0.28 \dim(\mathcal{X})$; and for our β-ResNet-VAE, $\dim(\mathcal{Z}) \approx 0.66 \dim(\mathcal{X})$. We did not observe improved results by simply increasing the number of latent channels/dimensions in the former model. We follow the same model training practice as reported by Minnen and Singh [2020], training both models to around 5 million gradient steps.

In our problem setup, we consider the reproduction alphabet to be the discrete set of pixel values $\mathcal{X} = \{0, 1, ..., 255\}^n$, so the decoder network ω needs to discretize the output of the final convolutional layer (in `Tensorflow`, this can be implemented by a `saturate_cast` to `uint8`). However, since the discretization operation has no useful gradient, we skip it during training, and only apply it for evaluation.

[4] https://github.com/tensorflow/compression/blob/f9edc949fa58381ffafa5aa8cb37dc8c3ce50e7f/models/ms2020.py

The operational quality-rate curves of various compression methods are taken from the `tensorflow-compression`[5] and `compressAI`[6] libraries. The results of Minnen and Singh [2020] represent the current state-of-the-art of neural image compression, to the best of our knowledge.

Lower Bound Experiments When running our lower bound algorithm on 256×256 crops of natural images, we observed similar behavior as on high-dimension Gaussians, with the loss $\tilde{\ell}_k$ quickly plateauing to a value around $\log k$. This issue persisted when we tried different u model architectures. See a discussion on this phenomenon in Section C.5.2.

C.6 Measures of Variability

As alluded to in Sec. 6.3, the proposed R-D bounds are estimated as empirical averages from samples, which only equal the true quantities in expectation. Before we characterize the variability of the bound estimates, we first define the exact form of the estimators used.

Upper Bound Estimator For the proposed upper bound, consider the variational problem solved by the Lagrangian Eq. 6.3, and define the random variables from the rate and distortion terms of the Lagrangian:

$$\mathcal{R}(Z, X) := \log q(Z|X) - \log q(Z),$$

[5] https://github.com/tensorflow/compression/tree/8692f3c1938b18f123dbd6e302503a23ce75330c/results/image_compression

[6] https://github.com/InterDigitalInc/CompressAI/tree/baca8e9ff070c9f712bf4206b8f2da942a0e3dfe/results

and

$$\mathcal{D}(Z, X) := \rho(X, \omega(Z)).$$

Above, X and Z follow the joint distribution $P_X Q_{Z|X}$, ω is the decoder function, and $q(z|x)$ and $q(z)$ are the Lebesgue density functions of absolutely continuous $Q_{Z|X=x}$ and Q_Z variational distributions (we only worked with absolutely continuous variational distributions for simplicity). Then it is clear that the expectations of the two random variables equal the rate and distortion terms, i.e.,

$$\mathbb{E}[\mathcal{R}(Z, X)] = \mathbb{E}_{x \sim P_X}[KL(Q_{Z|X=x} \| Q_Z)],$$

and

$$\mathbb{E}[\mathcal{D}(Z, X)] = \mathbb{E}_{P_X Q_{Z|X}}[\rho(X, \omega(Z))].$$

Recall for any given decoder and variational distributions, the above rate and distortion terms define a point $(\mathbb{E}[\mathcal{D}], \mathbb{E}[\mathcal{R}])$ that lies on an upper bound $R_U(D)$ of the source $R(D)$ curve. Given m i.i.d. pairs of $(Z_1, X_1), (Z_2, X_2), ..., (Z_m, X_m) \sim P_X Q_{Z|X}$, we estimate such a point by $(\mu_{\mathcal{D}}, \mu_{\mathcal{R}})$, using the usual sample mean estimators $\mu_{\mathcal{R}} := \frac{1}{m} \sum_{j=1}^{m} \mathcal{R}(Z_j, X_j)$ and $\mu_{\mathcal{D}} := \frac{1}{m} \sum_{j=1}^{m} \mathcal{D}(Z_j, X_j)$.

Lower Bound Estimator For the proposed lower bound, given i.i.d. $X_1, ..., X_k \sim P_X$, define the random variable

$$\xi := -\frac{1}{k} \sum_{i}^{k} \log u(X_i) - \frac{C_k}{\alpha} - \log \alpha + 1,$$

where the various quantities are defined as in Sec. 6.4. Then it is clear that its expectation equals the lower bound objective Eq. 6.8, i.e., $\mathbb{E}[\xi] = \tilde{\ell}_k := \mathbb{E}[-\log u(X)] - \mathbb{E}[C_k]/\alpha - \log \alpha + 1$. Given m i.i.d. samples of ξ, we form the usual sample mean estimator $\mu_\xi := \frac{1}{m}\sum_j \xi_j$, and form an R-D lower bound estimate by

$$\hat{R}_L(D) := -\lambda D + \mu_\xi.$$

In computing μ_ξ, we fix the constant α to a sample mean estimate of $\mathbb{E}[C_k]$ computed separately (as in line 7 of Algorithm 2). As explained in Sec. C.4, we repeat this procedure with different λ, and our overall R-D lower bound is obtained by taking the point-wise maximum of such linear under-estimators over λ.

Results Below we give measures of variability of for the sandwich bound estimates obtained on GAN generated images (Section 6.6.3). We report detailed statistics in the following tables, and plot them in Figure C.6. Our results on other experiments have comparable variability.

For the upper bound, given each trained β-VAE, we compute m samples of $(\mathcal{R}, \mathcal{D})$, essentially passing m samples of X through the autoencoder and collect the rate and distortion values; then we report the sample mean (μ), sample standard deviation (s), and 95% large-sample confidence interval for the population mean (the true expected value) for $\mathcal{R}$ and $\mathcal{D}$, respectively, in Tables C.1 and C.2.

For the lower bound, given each trained $\log u$ model, we compute m samples of ξ, then report the sample mean (μ_ξ), sample standard deviation (s_ξ), and 90% large-sample confidence lower bound for the population mean (the true expected value $\mathbb{E}[\xi] = \tilde{\ell}_k$) in Tables C.3 and C.4.

λ	$\mu_{\mathcal{R}}$	$s_{\mathcal{R}}$	$\mathbb{E}[\mathcal{R}]$ CI	$\mu_{\mathcal{D}}$	$s_{\mathcal{D}}$	$\mathbb{E}[\mathcal{D}]$ CI
300	0.278	0.8	(0.15, 0.41)	0.006	0.00287	(0.00553, 0.00647)
500	1.25	1.27	(1.04, 1.46)	0.00351	0.00207	(0.00317, 0.00385)
1e+03	2.69	1.21	(2.49, 2.89)	0.00157	0.00103	(0.0014, 0.00174)
4e+03	4.77	1.25	(4.56, 4.97)	0.000352	0.000285	(0.000305, 0.000399)
8e+03	5.65	1.31	(5.44, 5.87)	0.000196	0.000152	(0.000171, 0.000221)
1.6e+04	6.43	1.32	(6.21, 6.64)	0.000126	9.68e-05	(0.00011, 0.000142)

Table C.1: Statistics for the estimated R-D upper bound on GAN-generated images with $d = 2$, computed with $m = 100$ samples of $(\mathcal{R}, \mathcal{D})$. "CI" denotes 95% large-sample confidence interval.

λ	$\mu_{\mathcal{R}}$	$s_{\mathcal{R}}$	$\mathbb{E}[\mathcal{R}]$ CI	$\mu_{\mathcal{D}}$	$s_{\mathcal{D}}$	$\mathbb{E}[\mathcal{D}]$ CI
300	0.265	0.689	(0.15, 0.38)	0.00978	0.00376	(0.00916, 0.0104)
500	1.44	1.24	(1.23, 1.64)	0.00693	0.00284	(0.00646, 0.00739)
1e+03	4.08	1.73	(3.80, 4.36)	0.00332	0.00152	(0.00307, 0.00357)
2e+03	6.59	1.88	(6.28, 6.90)	0.00151	0.000752	(0.00139, 0.00164)
4e+03	8.66	1.89	(8.35, 8.97)	0.000774	0.000389	(0.00071, 0.000838)
8e+03	10.4	1.89	(10.06, 10.68)	0.000476	0.000229	(0.000438, 0.000514)
1.6e+04	12	1.83	(11.69, 12.29)	0.000323	0.000147	(0.000299, 0.000347)

Table C.2: Statistics for the estimated R-D upper bound on GAN-generated images with $d = 4$, computed with $m = 100$ samples of $(\mathcal{R}, \mathcal{D})$. "CI" denotes 95% large-sample confidence interval.

λ	μ_ξ	s_ξ	$\mathbb{E}[\xi]$ LCB
99.63	0.74	0.01	0.74
199.90	1.46	0.01	1.45
299.15	2.04	0.02	2.03
499.69	2.78	0.02	2.78
999.01	3.84	0.03	3.83
1999.70	4.58	0.05	4.57
2024.00	4.60	0.06	4.58
2999.30	4.97	0.05	4.95
3036.80	4.96	0.08	4.93
4000.00	5.18	0.05	5.17

Table C.3: Statistics for the estimated R-D lower bound on $d = 2$ GAN-generated images, computed with $m = 30$ samples of ξ. "LCB" denotes 90% large-sample lower confidence bound.

λ	μ_ξ	s_ξ	$\mathbb{E}[\xi]$ LCB
99.63	1.09	0.01	1.09
199.90	2.18	0.02	2.17
299.15	3.08	0.02	3.07
499.69	4.44	0.04	4.43
999.01	5.97	0.06	5.95
1999.70	6.69	0.08	6.67
2024.00	6.69	0.07	6.66
2999.30	6.87	0.08	6.84
3036.80	6.85	0.09	6.83
4000.00	6.93	0.08	6.91

Table C.4: Statistics for the estimated R-D lower bound on $d = 4$ GAN-generated images, computed with $m = 30$ samples of ξ. "LCB" denotes 90% large-sample lower confidence bound.

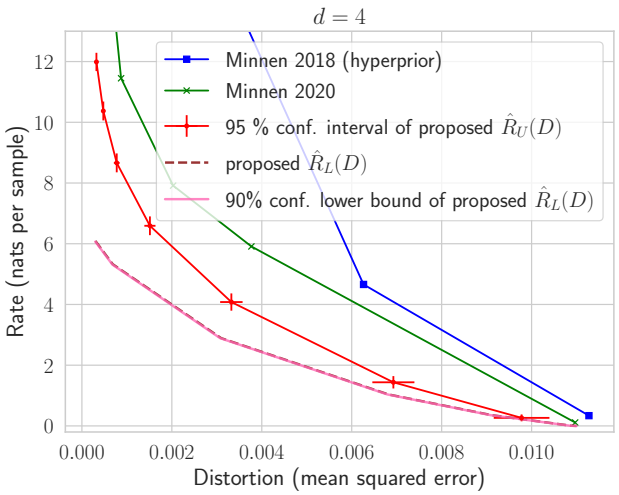


Figure C.6: Zoomed-in version of Figure 6.6-Middle-Bottom (GAN-generated images with intrinsic dimension $d = 4$), showing the sample mean estimates of R-D upper bound points $(\mu_\mathcal{D}, \mu_\mathcal{R})$ bracketed by 95% confidence intervals (**red**), as well as lower bound estimates based on the sample mean μ_ξ (**maroon**, dashed) and its 90% confidence lower bound (**pink** line, beneath **maroon**).

Appendix D

For Chapter 7, "Statistical and Optimal-Transport Perspectives on the Estimation of the Rate-Distortion Function"

We review probability theory background and explain our notation from the main text in Section D.1, give the formulas we used for numerically estimating an $R(D)$ upper bound in Section D.2, provide additional discussions and proofs regarding Wasserstein gradient descent in Section D.3, elaborate on the connections between the R-D estimation problem and variational inference/learning in Section D.4, provide additional experimental results and details in Section D.5, and list an example implementation of WGD in Section D.6.

D.1 Notions from probability theory

In this section we collect notions of probability theory used in the main text. See, e.g., [Çinlar, 2011] or [Folland, 1999] for more background.

Marginal and conditional distributions. The source and reproduction spaces $\mathcal{X}, \mathcal{Y}$ are equipped with sigma-algebras $\mathcal{A}_\mathcal{X}$ and $\mathcal{A}_\mathcal{Y}$, respectively. Let $\mathcal{X} \times \mathcal{Y}$ denote the product space equipped with the product sigma algebra $\mathcal{A}_\mathcal{X} \otimes \mathcal{A}_\mathcal{Y}$. For any probability measure π on $\mathcal{X} \times \mathcal{Y}$, its first **marginal** is

$$\pi_1(A) := \pi(A \times \mathcal{Y}), \quad A \in \mathcal{A}_\mathcal{X},$$

which is a probability measure on $\mathcal{X}$. When π is the distribution of a random vector (X, Y), then π_1 is the distribution of X. The second marginal of π is defined analogously as

$$\pi_2(B) := \pi(\mathcal{X} \times B), \quad B \in \mathcal{A}_\mathcal{Y}.$$

A Markov **kernel** or **conditional distribution** $K(x, dy)$ is a map $\mathcal{X} \times \mathcal{A}_\mathcal{Y} \to [0, 1]$ such that

1. $K(x, \cdot)$ is a probability measure on $\mathcal{Y}$ for each $x \in \mathcal{X}$;
2. the function $x \mapsto K(x, B)$ is measurable for each set $B \in \mathcal{A}_\mathcal{Y}$.

When speaking of the conditional distribution of a random variable Y given another random variable X, we occasionally also use the notation $Q_{Y|X}$ from information theory [Polyanskiy and Wu, 2022]. Then, $Q_{Y|X=x}(B) = K(x, B)$ is the conditional probability of the event $\{Y \in B\}$ given $X = x$.

Suppose that a probability measure μ on $\mathcal{X}$ is given, in addition to a kernel $K(x, dy)$.

Together they define a unique measure $\mu \otimes K$ on the product space $\mathcal{X} \times \mathcal{Y}$. For a rectangle set $A \times B \in \mathcal{A}_{\mathcal{X}} \otimes \mathcal{A}_{\mathcal{Y}}$,

$$\mu \otimes K(A \times B) = \int_A \mu(dx) K(x, B), \quad A \in \mathcal{A}_{\mathcal{X}}, B \in \mathcal{A}_{\mathcal{Y}}.$$

The measure $\pi := \mu \otimes K$ has first marginal $\pi_1 = \mu$.

The classic product measure is a special case of this construction. Namely, when a measure ν on $\mathcal{Y}$ is given, using the constant kernel $K(x, dy) := \nu(dy)$ (which does not depend on x) gives rise to the product measure $\mu \otimes \nu$,

$$\mu \otimes \nu(A \times B) = \mu(A)\nu(B), \quad A \in \mathcal{A}_{\mathcal{X}}, B \in \mathcal{A}_{\mathcal{Y}}.$$

Under mild conditions (for instance when $\mathcal{X}, \mathcal{Y}$ are Polish spaces equipped with their Borel sigma algebras, as in the main text), any probability measure π on $\mathcal{X} \times \mathcal{Y}$ is of the above form. Namely, the **disintegration** theorem asserts that π can be written as $\pi = \pi_1 \otimes K$ for some kernel K. When π is the joint distribution of a random vector (X, Y), this says that there is a measurable version of the conditional distribution $Q_{Y|X}$.

Optimal transport. Given a measure μ on $\mathcal{X}$ and a measurable function $T : \mathcal{X} \to \mathcal{Y}$, the **pushforward** (or image measure) of μ under T is a measure on $\mathcal{Y}$, given by

$$T_{\#}\mu(B) = \mu(T^{-1}(B)), \quad B \in \mathcal{A}_{\mathcal{Y}}.$$

If T is seen as a random variable and μ as the baseline probability measure, then $T_{\#}\mu$ is simply the distribution of T.

Suppose that μ and ν are probability measures on $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$ with finite second moment.

As introduced in the main text, $\Pi(\mu, \nu)$ denotes the set of couplings, i.e., measures π on $\mathcal{X} \times \mathcal{Y}$ with $\pi_1 = \mu$ and $\pi_2 = \nu$. The 2-Wasserstein distance $W_2(\mu, \nu)$ between μ and ν is defined as

$$W_2(\mu, \nu) = \left(\inf_{\pi \in \Pi(\mu,\nu)} \int \|y - x\|^2 \pi(dx, dy) \right)^{1/2}.$$

This indeed defines a metric on the space of probability measures with finite second moment.

We finish this section by giving a proof of the basic equivalence between optimization of $\mathcal{L}_{EOT}$ and $\mathcal{L}_{BA}$, which goes back to [Csiszár, 1974a, Lemma 1.3 and subsequent discussion]. Recall that $\Pi(\mu, \cdot)$ denotes the set of measures on the product space $\mathcal{X} \times \mathcal{Y}$ with $\pi_1 = \mu$.

Lemma D.1.1. *Set $\epsilon = \lambda^{-1}$. It holds that*

$$\inf_{\nu \in \mathcal{P}(\mathcal{Y})} \mathcal{L}_{EOT}(\nu) = \inf_{\nu \in \mathcal{P}(\mathcal{Y})} \lambda^{-1} \mathcal{L}_{BA}(\nu) \qquad \textit{and} \qquad \arg\min_{\nu \in \mathcal{P}(\mathcal{Y})} \mathcal{L}_{EOT}(\nu) = \arg\min_{\nu \in \mathcal{P}(\mathcal{Y})} \mathcal{L}_{BA}(\nu).$$

Proof. Both statements will follow from a simple property of relative entropy [Polyanskiy and Wu, 2022, Theorem 4.1, "golden formula"]: For $\pi \in \Pi(\mu, \cdot)$ with $H(\pi_2 \mid \nu) < \infty$, the properties of the logarithm reveal

$$H(\pi \mid \mu \otimes \nu) = H(\pi \mid \mu \otimes \pi_2) + H(\pi_2 \mid \nu), \tag{D.1}$$

and hence $H(\pi \mid \mu \otimes \nu) \geq H(\pi \mid \mu \otimes \pi_2)$. This implies

$$\begin{aligned}
\inf_{\nu \in \mathcal{P}(\mathcal{Y})} \mathcal{L}_{EOT}(\nu) &= \inf_{\pi \in \Pi(\mu, \cdot)} \int \rho d\pi + \epsilon H(\pi \mid \mu \otimes \pi_2) \\
&= \inf_{\pi \in \Pi(\mu, \cdot)} \inf_{\nu \in \mathcal{P}(\mathcal{Y})} \int \rho d\pi + \epsilon H(\pi \mid \mu \otimes \nu) \\
&= \frac{1}{\lambda} \inf_{\nu \in \mathcal{P}(\mathcal{Y})} \mathcal{L}_{BA}(\nu).
\end{aligned}$$

Further, any optimizer ν^* of $\mathcal{L}_{BA}$ with corresponding optimal "coupling" $\pi^* \in \Pi(\mu, \cdot)$ must

satisfy $H(\pi_2^* \mid \nu^*) = 0$ (otherwise, taking $\nu^* = \pi_2^*$ has better objective) and thus $\pi_2^* = \nu^*$ and $\pi^* \in \Pi(\mu, \nu^*)$, therefore ν^* is also an optimizer of $\mathcal{L}_{EOT}$ by the above equality. Conversely, any optimizer ν^* of $\mathcal{L}_{EOT}$ with coupling $\pi^* \in \Pi(\mu, \nu^*) \subset \Pi(\mu, \cdot)$ clearly yields a feasible solution for $\mathcal{L}_{BA}$ as well, and hence is also an optimizer of $\mathcal{L}_{BA}$ by the above equality. □

D.2 Numerical estimation of rate and distortion from a reproduction distribution

Given a reproduction distribution ν and a kernel $\mathcal{X} \times \mathcal{A}_\mathcal{Y} \to [0, 1]$, the tuple $(\mathcal{D}, \mathcal{R}) \in \mathbb{R}^2$ defined by

$$\mathcal{D} := \int \rho d(\mu \otimes K)$$

and

$$\mathcal{R} := H(\mu \otimes K | \mu \otimes \nu)$$

lies above the $R(D)$ curve. This again follows from the variational inequality (D.1) $\mathcal{R} \geq H(\pi|\mu \times (\mu \times K)_2) = I(\mu \times K)$, so that $\mathcal{R} \geq R(\mathcal{D}) =: \inf_{\pi \in \Pi(\pi, \cdot): \pi(\rho) \leq \mathcal{D}} I(\pi)$.

Given only a reproduction distribution ν, we will construct a kernel K from ν and use (μ, K) to compute $(\mathcal{D}, \mathcal{R})$, letting

$$\frac{dK(x, \cdot)}{d\nu}(y) = \frac{e^{-\lambda \rho(x,y)}}{\int e^{-\lambda \rho(x, \tilde{y})} \nu(d\tilde{y})}$$

as in the first step of the BA algorithm. This choice of K is in fact optimal as it achieves the minimum in the definition of $\mathcal{L}_{BA}$ (7.5) for the given ν. Plugging K into the formulas for $\mathcal{D}$

and $\mathcal{R}$ gives

$$\mathcal{D} = \int \rho(x,y) e^{\varphi(x) - \lambda\rho(x,y)} \mu \otimes \nu(dx, dy) \tag{D.2}$$

and

$$\begin{aligned} \mathcal{R} &= \int \int \log\left(\frac{K(x,dy)}{\nu(dy)}\right) K(x,dy)\mu(dx) \\ &= \int [\varphi(x) - \lambda\rho(x,y)] e^{\varphi(x) - \lambda\rho(x,y)} \mu \otimes \nu(dx,dy), \end{aligned} \tag{D.3}$$

where we introduced the shorthand

$$\varphi(x) := -\log \int_{\mathcal{Y}} e^{-\lambda\rho(x,y)} \nu(dy). \tag{D.4}$$

Note that we have the following relation (which explains (7.6))

$$\mathcal{L}_{BA}(\nu) = \mathcal{R} + \lambda\mathcal{D} = \int \varphi(x)\mu(dx).$$

Let ν be an n-point measure, $\nu = \sum_{i=1}^n w_i \delta_{y_i}$, e.g., the output of WGD or in the inner step of NERD. Then $\varphi(x)$ can be evaluated exactly as a finite sum over the ν particles, and the expressions above for $\mathcal{D}$ and $\mathcal{R}$ (which are integrals w.r.t. the product distribution $\mu \otimes \nu$) can be estimated as sample averages. That is, given m independent samples $\{x_i\}_{i=1}^m$ from μ, we compute unbiased estimates

$$\begin{aligned} \hat{\mathcal{D}} &= \sum_{i=1}^m \sum_{j=1}^n \frac{1}{m} w_j \rho(x_i, y_j) e^{\varphi(x_i) - \lambda\rho(x_i,y_j)}, \\ \hat{\mathcal{R}} &= \sum_{i=1}^m \sum_{j=1}^n \frac{1}{m} w_j [\varphi(x_i) - \lambda\rho(x_i, y_j)] e^{\varphi(x_i) - \lambda\rho(x_i,y_j)}, \end{aligned}$$

where $\varphi(x) = -\log \sum_{j=1}^n e^{\varphi(x_i) - \lambda\rho(x_i,y_j)}$. Similarly, a sample mean estimate for $\mathcal{L}_{BA}$ is given

by

$$\hat{\mathcal{L}}_{BA}(\nu) = \frac{1}{m}\sum_{i=1}^{m}\varphi(x_i) = -\frac{1}{m}\sum_{i=1}^{m}\log\left(\sum_{j=1}^{n} e^{\varphi(x_i)-\lambda\rho(x_i,y_j)}\right). \tag{D.5}$$

In practice, we found it simpler and numerically more stable to instead compute $\hat{\mathcal{R}}$ as

$$\hat{\mathcal{R}} = \hat{\mathcal{L}}_{BA}(\nu) - \lambda\hat{\mathcal{D}}.$$

Whenever possible, we avoid exponentiation and instead use `logsumexp` to prevent numerical issues.

D.3 Wasserstein gradient descent

D.3.1 On Wasserstein gradients of the EOT and rate functionals

First, we elaborate on the Wasserstein gradient of the EOT functional $\mathcal{L}_{EOT}(\nu)$. That the dual potential from Sinkhorn's algorithm is differentiable follows from the fact that optimal dual potentials satisfy the Schrödinger equations (cf. [Nutz, 2021, Corollary 2.5]). Differentiability was shown in [Genevay et al., 2019, Theorem 2] in the compact setting, and in [Mena and Niles-Weed, 2019, Proposition 1] in unbounded settings. While Mena and Niles-Weed [2019] only states the result for quadratic cost, the approach of Proposition 1 therein applies more generally.

Below, we compute the Wasserstein gradient of the rate functional $\mathcal{L}_{BA}(\nu)$. Recall from (7.6),

$$\mathcal{L}_{BA}(\nu) = \int -\log\int \exp(-\lambda\rho(x,y))\nu(dy)\mu(dx).$$

Under sufficient integrability on μ and ν to exchange the order of limit and integral, we can calculate the first variation as

$$\begin{aligned}\lim_{\varepsilon\to 0}\frac{\mathcal{L}((1-\varepsilon)\nu+\varepsilon\tilde{\nu})-\mathcal{L}(\nu)}{\varepsilon} &= -\int \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\log\left[\frac{\int\exp(-\lambda\rho(x,y))(\nu+\varepsilon(\tilde{\nu}-\nu))(dy)}{\int\exp(-\lambda\rho(x,y))\nu(dy)}\right]\mu(dx)\\ &= -\int \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\log\left[1+\frac{\int\exp(-\lambda\rho(x,y))\varepsilon(\tilde{\nu}-\nu)(dy)}{\int\exp(-\lambda\rho(x,y))\nu(dy)}\right]\mu(dx)\\ &= \iint -\frac{\exp(-\lambda\rho(x,y))}{\int\exp(-\lambda\rho(x,\tilde{y}))\nu(d\tilde{y})}\mu(dx)\,(\tilde{\nu}-\nu)(dy),\end{aligned}$$

where the last equality uses $\lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\log(1+\varepsilon x)=x$ and Fubini's theorem. Thus the first variation ψ^ν of $\mathcal{L}_{BA}$ at ν is

$$\psi^\nu(y)=\int -\frac{\exp(-\lambda\rho(x,y))}{\int\exp(-\lambda\rho(x,\tilde{y}))\nu(d\tilde{y})}\mu(dx). \tag{D.6}$$

To find the desired Wasserstein gradient of $\mathcal{L}_{BA}$, it remains to take the Euclidean gradient of ψ^ν, i.e., $\nabla\mathcal{L}_{BA}(\nu)=\nabla\psi^\nu$. Doing so gives us the desired Wasserstein gradient:

$$\begin{aligned}\nabla V_{\mathcal{L}_{BA}}(\nu)[y] &= \nabla_y\psi^\nu(y)\\ &= \frac{\partial}{\partial y}\int -\frac{\exp(-\lambda\rho(x,y))}{\int\exp(-\lambda\rho(x,\tilde{y}))\nu(d\tilde{y})}\mu(dx)\\ &= \int\frac{\exp(-\lambda\rho(x,y))\lambda\frac{\partial}{\partial y}\rho(x,y)}{\int\exp(-\lambda\rho(x,\tilde{y}))\nu(d\tilde{y})}\mu(dx),\end{aligned} \tag{D.7}$$

again assuming suitable regularity conditions on ρ and μ to exchange the order of integral and differentiation.

D.3.2 Proof of Proposition 7.4.2 (convergence of Wasserstein gradient descent)

We first provide an auxiliary result.

Lemma D.3.1. *Let* $\gamma_1 \geq \gamma_2 \geq \cdots \geq 0$ *and* $a_t \geq 0$, $t \in \mathbb{N}$, $C > 0$ *satisfy* $\sum_{t=1}^{\infty} \gamma_t = \infty$, $\sum_{t=1}^{\infty} \gamma_t^2 < \infty$, $\sum_{t=1}^{\infty} a_t \gamma_t < \infty$ *and* $|a_t - a_{t+1}| \leq C\gamma_t$ *for all* $t \in \mathbb{N}$. *Then* $\lim_{t\to\infty} a_t = 0$.

Proof. The conclusion remains unchanged when rescaling a_t by the constant C, and thus without loss of generality $C = 1$.

Clearly $\gamma_t \to 0$ as $\sum_{t=1}^{\infty} \gamma_t^2 < \infty$. Moreover, there exists a subsequence of $(a_t)_{t\in\mathbb{N}}$ which converges to zero (otherwise there exists $\delta > 0$ such that $a_t \geq \delta > 0$ for all but finitely many t, contradicting $\sum_{t=1}^{\infty} \gamma_t a_t < \infty$).

Arguing by contradiction, suppose that the conclusion fails, i.e., that there exists a subsequence of $(a_t)_{t\in\mathbb{N}}$ which is uniformly bounded away from zero, say $a_t \geq \delta > 0$ along that subsequence. Using this subsequence and the convergent subsequence mentioned above, we can construct a subsequence $a_{i_1}, a_{i_2}, a_{i_3}, \ldots$ where $a_{i_n} \approx 0$ for n odd and $a_{i_n} \geq \delta$ for n even. We will show that

$$\sum_{t=i_{2n-1}}^{i_{2n}} a_t \gamma_t \gtrsim \delta^2/2 \qquad \text{for all } n \in \mathbb{N},$$

contradicting the finiteness of $\sum_t \gamma_t a_t$. (The notation $\approx$ ($\gtrsim$) indicates (in)equality up to additive terms converging to zero for $n \to \infty$.)

To ease notation, fix n and set $m = i_{2n-1}$ and $M = i_{2n}$. We show that $\sum_{t=m}^{M} a_t \gamma_t \gtrsim \delta^2/2$. To this end, using $|a_t - a_{t+1}| \leq \gamma_t$ we find

$$a_t \geq a_M - \sum_{j=k}^{M-1} \gamma_j \geq \delta - \sum_{j=k}^{M-1} \gamma_j.$$

Since $a_m \approx 0$, there exists a largest $n_0 \in \mathbb{N}$, $n_0 \geq m$, such that $\sum_{j=n_0}^{M-1} \gamma_j \gtrsim \delta$ (and thus

$\sum_{j=n_0}^{M-1} \gamma_j \lesssim \delta - \gamma_{n_0} \approx \delta$ as well). We conclude

$$\sum_{t=m}^{M} \gamma_t a_t \geq \sum_{t=n_0}^{M} \gamma_t a_t \geq \sum_{t=n_0}^{M} \gamma_t \left(\delta - \sum_{j=k}^{M-1} \gamma_j \right) \gtrsim \delta^2 - \sum_{t=n_0}^{M} \sum_{j=n_0}^{M} \gamma_t \gamma_j \mathbf{1}_{\{j \geq k\}}$$
$$= \delta^2 - \frac{1}{2} \left(\sum_{t=n_0}^{M} \gamma_t \right)^2 - \frac{1}{2} \sum_{t=n_0}^{M} \gamma_t^2 \approx \delta^2/2,$$

where we used that $\sum_{t=n_0}^{M} \gamma_t^2 \approx 0$. This completes the proof. □

Proof of Proposition 7.4.2. Using the linear approximation property in (7.14), we calculate

$$\mathcal{L}(\nu^{(n)}) - \mathcal{L}(\nu^{(0)}) = \sum_{t=0}^{n-1} \mathcal{L}(\nu^{(t+1)}) - \mathcal{L}(\nu^{(t)})$$
$$= \sum_{t=0}^{n-1} -\gamma_t \int \|\nabla V_{\mathcal{L}}(\nu^{(t)})\|^2 \, d\nu^{(t)} + \gamma_t^2 \, o\left(\int \|\nabla V_{\mathcal{L}}(\nu^{(t)})\|^2 \, d\nu^{(t)} \right).$$

As $\mathcal{L}(\nu^{(0)})$ is finite and $\mathcal{L}(\nu^{(n)})$ is bounded from below, it follows that

$$\sum_{t=0}^{\infty} \gamma_t \int \|\nabla V_{\mathcal{L}}(\nu^{(t)})\|^2 \, d\nu^{(t)} < \infty.$$

The claim now follow by applying Lemma D.3.1 with $a_t = \int \|\nabla \psi^{\nu^{(t)}}\|^2 \, d\nu^{(t)}$; note that the assumption in the lemma, $|a_t - a_{t+1}| \leq C\gamma_t$, is satisfied due to the second inequality in (7.14)

and a short calculation below

$$\begin{aligned}\left|\int \|\nabla\psi^{\nu^{(t)}}\|^2 - \int \|\nabla\psi^{\nu^{(t+1)}}\|^2\right| &\le CW_2(\nu^{(t)},\nu^{(t+1)}) \\ &\le C\left(\int\int \|x-y\|^2 \delta_{x-\gamma_t\nabla V_{\mathcal{L}}(\nu^{(t)})[x]}(dy)\nu^{(t)}(dx)\right)^{\frac{1}{2}} \\ &\le C\gamma_t\left(\int \|\nabla V_{\mathcal{L}}(\nu^{(t)})\|^2\nu^{(t)}(dx)\right)^{\frac{1}{2}} \\ &\le C\gamma_t \sup_{t'}\left(\int \|\nabla V_{\mathcal{L}}(\nu^{(t')})\|^2\nu^{(t')}(dx)\right)^{\frac{1}{2}} \\ &\le C'\gamma_t,\end{aligned}$$

where we use $\sup_t \int \|\nabla V_{\mathcal{L}}(\nu^{(t)})\|^2\nu^{(t)}(dx) < \infty$ in the last step. □

D.3.3 Proof of Proposition 7.4.3 (sample complexity)

Recall that $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$ and $\rho(x,y) = \|x-y\|^2$ in this proposition. For the proof, we will need the following lemma which is of independent interest. We write $\nu \le_c \mu$ if ν is dominated by μ in convex order, i.e., $\int f\,d\nu \le \int f\,d\mu$ for all convex functions $f : \mathbb{R}^d \to \mathbb{R}$.

Lemma D.3.2. *Let μ have finite second moment. Given $\nu \in \mathcal{P}(\mathbb{R}^d)$, there exists $\tilde{\nu} \in \mathcal{P}(\mathbb{R}^d)$ with $\tilde{\nu} \le_c \mu$ and*

$$\mathcal{L}_{EOT}(\mu,\tilde{\nu}) \le \mathcal{L}_{EOT}(\mu,\nu).$$

This inequality is strict if $\nu \not\le_c \mu$. In particular, any optimizer ν^* of (7.8) satisfies $\nu^* \le_c \mu$.

Proof. Because this proof uses disintegration over $\mathcal{Y}$, it is convenient to reverse the order of the spaces in the notation and write a generic point as $(x,y) \in \mathcal{Y}\times\mathcal{X}$. Consider $\pi \in \Pi(\nu,\mu)$

and its disintegration $\pi = \nu(dx) \otimes K(x, dy)$ over $x \in \mathcal{Y}$. Define $T : \mathbb{R}^d \to \mathbb{R}^d$ by

$$T(x) := \int y \, K(x, dy).$$

Define also $\tilde{\pi} := (T, \mathrm{id})_{\#}\pi$ and $\tilde{\nu} := \tilde{\pi}_1$. From the definition of T, we see that $\tilde{\pi}$ is a martingale, thus $\tilde{\nu} \leq_c \mu$. Moreover, $\tilde{\nu} \otimes \mu = (T, \mathrm{id})_{\#}\nu \otimes \mu$. The data-processing inequality now shows that

$$H(\tilde{\pi}|\tilde{\nu} \otimes \mu) \leq H(\pi|\nu \otimes \mu).$$

On the other hand, $\int \| \int \tilde{y} \, K(x, d\tilde{y}) - y\|^2 \, K(x, dy) \leq \int \|x - y\|^2 K(x, dy)$ since the barycenter minimizes the squared distance, and this inequality is strict whenever $x \neq \int \tilde{y} K(x, d\tilde{y})$. Thus

$$\int \|x - y\|^2 \, \tilde{\pi}(dx, dy) \leq \int \|x - y\|^2 \, \pi(dx, dy),$$

and the inequality is strict unless $T(x) = x$ for ν-a.e. x, which in turn is equivalent to π being a martingale. The claims follow. □

Proof of Proposition 7.4.3. Subgaussianity of the optimizer follows directly from Lemma D.3.2.

Recalling that $\inf_\nu \mathcal{L}_{EOT}(\nu)$ and $\inf_\nu \lambda^{-1}\mathcal{L}_{BA}(\nu)$ have the same values and minimizers (given by (7.9) in Sec. 7.2.2), it suffices to show the claim for $\mathcal{L} = \mathcal{L}_{EOT}$. Let ν^* be an optimizer of (7.8) (i.e., an optimal reproduction distribution) and ν^n its empirical measure from n samples, then clearly

$$\begin{aligned}\left| \min_{\nu_n \in \mathcal{P}_n(\mathbb{R}^d)} \mathcal{L}_{EOT}(\mu, \nu_n) - \min_{\nu \in \mathcal{P}(\mathbb{R}^d)} \mathcal{L}_{EOT}(\mu, \nu) \right| &= \min_{\nu_n \in \mathcal{P}_n(\mathbb{R}^d)} \mathcal{L}_{EOT}(\mu, \nu_n) - \min_{\nu \in \mathcal{P}(\mathbb{R}^d)} \mathcal{L}_{EOT}(\mu, \nu) \\ &\leq \mathbb{E}\left[|\mathcal{L}_{EOT}(\mu, \nu^n) - \mathcal{L}_{EOT}(\mu, \nu^*)| \right]\end{aligned}$$

where the expectation is taken over samples for ν^n. The first inequality of Proposition 7.4.3 now follows from the sample complexity result for entropic optimal transport in [Mena and Niles-Weed, 2019, Theorem 2].

Denote by ν_m^* the optimizer for the problem (7.8) with μ replaced by μ^m. Similarly to the above, we obtain

$$\mathbb{E}\left[\left|\min_{\nu\in\mathcal{P}(\mathbb{R}^d)} \mathcal{L}_{EOT}(\mu,\nu) - \min_{\nu\in\mathcal{P}(\mathbb{R}^d)} \mathcal{L}_{EOT}(\mu^m,\nu)\right|\right] \leq \mathbb{E}\left[\max_{\nu\in\{\nu^*,\nu_m^*\}} |\mathcal{L}_{EOT}(\mu,\nu) - \mathcal{L}_{EOT}(\mu^m,\nu)|\right],$$

where the expectation is taken over samples from μ^m. In this situation, we cannot directly apply [Mena and Niles-Weed, 2019, Theorem 2]. However, the bound given by [Mena and Niles-Weed, 2019, Proposition 2] still applies, and the only dependence on $\nu \in \{\nu^*, \nu_m^*\}$ is through their subgaussianity constants. By Lemma D.3.2, these constants are bounded by the corresponding constants of μ and μ^m. Thus, the arguments in the proof of [Mena and Niles-Weed, 2019, Theorem 2] can be applied, yielding the second inequality of Proposition 7.4.3.

The final inequality of Proposition 7.4.3 follows from the first two inequalities (the first one being applied with μ^m) and the triangle inequality, where we again use the arguments in the proof of [Mena and Niles-Weed, 2019, Theorem 2] to bound the expectation over the subgaussianity constants of μ^m. □

D.3.4 Convergence of the proposed hybrid algorithm

In our present work, our hybrid algorithm targets the non-stochastic setting and is motivated by the fact that the BA update is in some sense orthogonal to the Wasserstein gradient update, and can only monotonically improve the objective. While empirically we observe the

additional BA steps to not hurt – but rather accelerate – the convergence of WGD (see 7.5.1), additional effort is required to theoretically guarantee convergence of the hybrid algorithm.

There are two key properties we use for the convergence of the base WGD algorithm: 1) a certain monotonicity of the update steps (up to higher order terms, gradient descent improves the objective) and 2) stability of gradients across iterations. If we include the BA step, we find that 1) still holds, but 2) may a-priori be lost. Indeed, 1) holds since BA updates monotonically improve the objective. Using just 1), we can still obtain a Pareto convergence of the gradients for the hybrid algorithm, $\sum_{t=0}^{\infty} \gamma_t \int \|\nabla V_{\mathcal{L}}(\nu^{(t)})\|^2 \, d\nu^{(t)} < \infty$ (here $\nu^{(t)}$ are the outputs from the respective BA steps and γ_t is the step size of the gradient steps). Without property 2), we cannot conclude $\int \|\nabla V_{\mathcal{L}}(\nu^{(t)})\|^2 \, d\nu^{(t)} \to 0$ for $t \to \infty$. We emphasize that in practice, it still appears that 2) holds even after including the BA step. Motivated by this analysis, an adjusted hybrid algorithm where, e.g., the BA update is rejected if it causes a drastic change in the Wasserstein gradient, could guarantee that 2) holds with little practical changes. From a different perspective, we also believe the hybrid algorithm may be tractable to study in relation to gradient flows in the Wasserstein-Fisher-Rao geometry (cf. [Yan et al., 2023]), in which the BA step corresponds to a gradient update in the Fisher-Rao geometry with a unit step size.

In the stochastic setting, the BA (and therefore our hybrid) algorithm does not directly apply, as performing BA updates on mini-batches no longer guarantees monotonic improvement of the overall objective. Extending the BA and hybrid algorithm to this setting would be interesting future work.

Table D.1: Guide to notation and their interpretations in various problem domains. "LVM" stands for latent variable modeling, "NPMLE" stands for non-parametric maximum-likelihood estimation. The R-D problem (7.3) is equivalent to a projection problem under an entropic optimal transport cost (discussed in Sec. 7.2.2) and statistical problems involving maximum-likelihood estimation (see discussion in Sec. 7.2.3 and below).

Context	$\mu = P_X$	$\rho(x, y)$	$K = Q_{Y\|X}$	$\nu = Q_Y$
OT	source distribution	transport cost	"transport plan"	target distribution
R-D	data distribution	distortion criterion	compression algorithm	codebook distribution
LVM/NPMLE	data distribution	"$-\log p(x\|y)$"	variational posterior	prior distribution
deconvolution	noisy measurements	"noise kernel"	—	noiseless model

D.4 R-D estimation and variational inference/learning

In this section, we elaborate on the connection between the R-D problem (7.3) and variational inference and learning in latent variable models, of which maximum likelihood deconvolution (discussed in Sec. 7.2.3) can be seen as a special case. Also see Section 3 of [Yang et al., 2023a] for a related discussion in the context of lossy data compression.

To make the connections clearer to a general machine learning audience, we adopt notation more common in statistics and information theory. Table D.1 summarizes the notation and the correspondence to the measure-theoretic notation used in the main text.

In statistics, a common goal is to model an (unknown) data generating distribution P_X by some density model $\hat{p}(x)$. In particular, here we will choose $\hat{p}(x)$ to be a latent variable model, where $\mathcal{Y}$ takes on the role of a latent space, and $Q_Y = \nu$ is the distribution of a latent variable Y (which may encapsulate the model parameters). As we shall see, the optimization objective defining the rate functional (7.5) corresponds to an aggregate Evidence LOwer Bound (ELBO) [Blei et al., 2017]. Thus, computing the rate functional corresponds to variational inference [Blei et al., 2017] in a given model (see Sec. D.4.2), and the parametric

R-D estimation problem, i.e.,

$$\inf_{\nu \in \mathcal{H}} \mathcal{L}_{BA}(\nu),$$

is equivalent to estimating a latent variable model using the variational EM algorithm [Beal and Ghahramani, 2003b] (see Sec. D.4.3). The variational EM algorithm can be seen as a restricted version of the BA algorithm (see Sec. D.4.3), whereas the EM algorithm [Dempster et al., 1977] shares its E-step with the BA algorithm but can differ in its M-step (see Sec. D.4.4).

D.4.1 Setup

For concreteness, fix a reference measure ζ on $\mathcal{Y}$, and suppose Q_Y has density $q(y)$ w.r.t. ζ. Often the latent space $\mathcal{Y}$ is a Euclidean space, and $q(y)$ is the usual probability density function w.r.t. the Lebesgue measure ζ; or when the latent space is discrete/countable, ζ is the counting measure and $q(y)$ is a probability mass function. We will consider the typical parametric estimation problem and choose a particular parametric form for Q_Y indexed by a parameter vector θ. This defines a parametric family $\mathcal{H} = \{Q_Y^\theta : \theta \in \Theta\}$ for some parameter space Θ. Finally, suppose the distortion function ρ induces a conditional likelihood density via $p(x|y) \propto e^{-\lambda \rho(x,y)}$, with a normalization constant that is constant in y.

These choices then result in a latent variable model specified by the joint density $q(y)p(x|y)$, and we model the data distribution with the marginal density: [1]

$$\hat{p}(x) = \int_{\mathcal{Y}} p(x|y) dQ_Y(y) = \int_{\mathcal{Y}} p(x|y) q(y) \zeta(dy). \tag{D.8}$$

As an example, a Gaussian mixture model with isotropic component variances can be specified

[1] To be more precise, we always fix a reference measure η on $\mathcal{X}$, so that densities such as $\hat{p}(x)$ and $p(x|y)$ are with respect to η. In the applications we considered in this work, η is the Lebesgue measure on $\mathcal{X} = \mathbb{R}^d$.

as follows. Let Q_Y be a mixing distribution on $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$ parameterized by component weights $w_{1,\dots,k}$ and locations $\mu_{1,\dots,k}$, such that $Q_Y = \sum_{k=1}^{K} w_k \delta_{\mu_k}$. Let $p(x|y) = \mathcal{N}(y, \sigma^2)$ be a conditional Gaussian density with mean y and variance σ^2. Now formula (D.8) gives the usual Gaussian mixture density on $\mathbb{R}^d$.

Maximum-likelihood estimation then ideally maximizes the population log (marginal) likelihood,

$$\mathbb{E}_{x\sim P_X}[\log \hat{p}(x)] = \int \log \hat{p}(x) P_X(dx) = \int \log \left(\int_{\mathcal{Y}} p(x|y) dQ_Y(y) \right) P_X(dx). \tag{D.9}$$

The maximum-likelihood deconvolution setup can be seen as a special case where the form of the marginal density (D.8) derives from knowledge of the true data generation process, with $P_X = \alpha * \mathcal{N}(0, \frac{1}{\lambda})$ for some unknown α and known noise $\mathcal{N}(0, \frac{1}{\lambda})$ (i.e., the model is well-specified). We note in passing that the idea of estimating an unknown data distribution by adding artificial noise to it also underlies work on spread divergence [Zhang et al., 2020] and denoising score matching [Vincent, 2011].

To deal with the often intractable marginal likelihood in the inner integral, we turn to variational inference and learning [Jordan et al., 1999, Wainwright et al., 2008].

D.4.2 Connection to variational inference

Given a latent variable model and any data observation x, a central task in Bayesian statistics is to infer the Bayesian posterior [Jordan, 1999], which we formally view as a conditional distribution $Q^*_{Y|X=x}$. It is given by

$$\frac{dQ^*_{Y|X=x}(y)}{dQ_Y(y)} = \frac{p(x|y)}{\hat{p}(x)},$$

or, in terms of the density $q(y)$ of Q_Y, given by the following conditional density via the familiar Bayes' rule,

$$q^*(y|x) = \frac{p(x|y)q(y)}{\hat{p}(x)} = \frac{p(x|y)q(y)}{\int_{\mathcal{Y}} p(x|y)q(y)\zeta(dy)}.$$

Unfortunately, the true Bayesian posterior is typically intractable, as the (marginal) data likelihood in the denominator involves an often high-dimensional integral. Variational inference [Jordan et al., 1999, Wainwright et al., 2008] therefore aims to approximate the true posterior by a variational distribution $Q_{Y|X=x} \in \mathcal{P}(\mathcal{Y})$ by minimizing their relative divergence $H(Q_{Y|X=x}|Q^*_{Y|X=x})$. The problem is equivalent to maximizing the following lower bound on the marginal log-likelihood, known as the Evidence Lower BOund (ELBO) [Blei et al., 2017]:

$$\begin{aligned}
\arg\min_{Q_{Y|X=x}} H(Q_{Y|X=x}|Q^*_{Y|X=x}) &= \arg\max_{Q_{Y|X=x}} \mathrm{ELBO}(Q_Y, x, Q_{Y|X=x}), \\
\mathrm{ELBO}(Q_Y, x, Q_{Y|X=x}) &= \mathbb{E}_{y\sim Q_{Y|X=x}}[\log p(x|y)] - H(Q_{Y|X=x}|Q_Y) \\
&= \log \hat{p}(x) - H(Q_{Y|X=x}|Q^*_{Y|X=x}).
\end{aligned} \tag{D.10}$$

Recall the rate functional (7.5) arises through an optimization problem over a transition kernel K,

$$\mathcal{L}_{BA}(\nu) = \inf_K \lambda \int \rho d(\mu \otimes K) + H(\mu \otimes K|\mu \otimes \nu).$$

Translating the above into the present notation ($\mu \to P_X, K \to Q_{Y|X}, \nu \to Q_Y$; see Table D.1), we obtain

$$\begin{aligned}
\mathcal{L}_{BA}(Q_Y) &= \inf_{Q_{Y|X}} \mathbb{E}_{x\sim P_X, y\sim Q_{Y|X=x}}[-\log p(x|y)] + \mathbb{E}_{x\sim P_X}[H(Q_{Y|X=x}|Q_Y)] + \text{const} \\
&= \inf_{Q_{Y|X}} \mathbb{E}_{x\sim P_X}[-\mathrm{ELBO}(Q_Y, x, Q_{Y|X=x})] + \text{const}.
\end{aligned} \tag{D.11}$$

which allows us to interpret the rate functional as the result of performing variational inference through ELBO optimization. At optimality, $Q_{Y|X} = Q^*_{Y|X}$, the ELBO (D.10) is tight and recovers $\log \hat{p}(x)$, and the rate functional takes on the form of a (negated) population marginal log likelihood (D.9), as given earlier by (7.6) in Sec. 7.2.1.

D.4.3 Connection to variational EM

The discussion so far concerns *probabilistic inference*, where a latent variable model $(Q_Y, p(x|y))$ has been given and we saw that computing the rate functional amounts to variational inference. Suppose now we wish to *learn* a model from data. The R-D problem (7.4) then corresponds to model estimation using the variational EM algorithm [Beal and Ghahramani, 2003b].

To estimate a latent variable model by (approximate) maximum-likelihood, the variational EM algorithm maximizes the population ELBO

$$\mathbb{E}_{x\sim P_X}[\mathrm{ELBO}(Q_Y, x, Q_{Y|X=x})] = \mathbb{E}_{x\sim P_X, y\sim Q_{Y|X=x}}[\log p(x|y)] - \mathbb{E}_{x\sim P_X}[H(Q_{Y|X=x}|Q_Y)], \tag{D.12}$$

w.r.t. Q_Y and $Q_{Y|X}$. This precisely corresponds to the R-D problem $\inf_{Q_Y\in\mathcal{H}} \mathcal{L}_{BA}(Q_Y)$, using the form of $\mathcal{L}_{BA}(Q_Y)$ from (D.11).

In popular implementations of variational EM such as the VAE [Kingma and Welling, 2013], Q_Y and $Q_{Y|X}$ are restricted to parametric families. When they are allowed to range over all of $\mathcal{P}(\mathcal{Y})$ and all conditional distributions, variational EM then becomes equivalent to the BA algorithm.

D.4.4 The Blahut–Arimoto and (exact) EM algorithms

Here we focus on the distinction between the BA algorithm and the (exact) EM algorithm [Dempster et al., 1977], rather than the *variational EM* algorithm. Both BA and (exact) EM perform coordinate descent on the same objective function (namely the negative of the population ELBO from (D.12)), but they define the coordinates slightly differently — the BA algorithm uses $(Q_{Y|X}, Q_Y)$ with $Q_Y \in \mathcal{P}(\mathcal{Y})$, whereas the EM algorithm uses $(Q_{Y|X}, \theta)$ with θ indexing a parametric family $\mathcal{H} = \{Q_Y^\theta : \theta \in \Theta\}$. Thus the coordinate update w.r.t. $Q_{Y|X}$ (the "E-step") is the same in both algorithms, but the subseuquent "M-step" potentially differs depending on the role of θ.

Given the optimization objective, which is simply the negative of (D.12),

$$\mathbb{E}_{x\sim P_X, y\sim Q_{Y|X=x}}[-\log p(x|y)] + H(P_X Q_{Y|X} | P_X \otimes Q_Y), \tag{D.13}$$

both BA and EM optimize the transition kernel $Q_{Y|X}$ the same way in the E-step, as

$$\frac{dQ^*_{Y|X=x}}{dQ_Y}(y) = \frac{p(x|y)}{\hat{p}(x)}. \tag{D.14}$$

For the M-step, the BA algorithm only minimizes the relative entropy term of the objective (D.13),

$$\min_{Q_Y \in \mathcal{P}(\mathcal{Y})} H(P_X Q^*_{Y|X}; P_X \otimes Q_Y),$$

(with the optimal Q_Y given by the second marginal of $P_X Q^*_{Y|X}$) whereas the EM algorithm minimizes the full objective w.r.t. the parameters θ of Q_Y,

$$\min_{\theta\in\Theta} \mathbb{E}_{(x,y)\sim P_X Q^*_{Y|X}}[-\log p(x|y)] + H(P_X Q^*_{Y|X}; P_X \otimes Q_Y). \tag{D.15}$$

The difference comes from the fact that when we parameterize Q_Y by θ in the parameter estimation problem, $Q^*_{Y|X}$ — and consequently both terms in the objective of (D.15) — will have functional dependence on θ through the E-step optimality condition (D.14).

In the Gaussian mixture example, $Q_Y = \sum_{k=1}^K w_k \delta_{\mu_k}$, and its parameters θ consist of the components weights $(w_1, ..., w_K) \in \Delta^{d-1}$ and location vectors $\{\mu_1, ..., \mu_K\} \subset \mathbb{R}^d$. The E-step computes $Q^*_{Y|X=x} = \sum_k w_k \frac{p(x|\mu_k)}{p(x)} \delta_{\mu_k}$. For the M-step, if we regard the locations as known so that $\theta = (w_1, ..., w_K)$ only consists of the weights, then the two algorithms perform the same update; however if θ also includes the locations, then the M-step of the EM algorithm will not only update the weights as in the BA algorithm, but also the locations, due to the distortion term $\mathbb{E}_{(x,y)\sim P_X Q^*_{Y|X}}[-\log p(x|y)] = -\int \sum_k w_k \frac{p(x|\mu_k)}{p(x)} \log p(x|\mu_k) P_X(dx)$.

D.5 Further experimental results

Our deconvolution experiments were run on Intel(R) Xeon(R) CPUs, and the rest of the experiments were run on Titan RTX GPUs.

In most experiments, we use the Adam [Kingma and Ba, 2015] optimizer for updating the ν particle locations in WGD and for updating the variational parameters in other methods. For our hybrid WGD algorithm, which adjusts the particle weights in addition to their locations, we found that applying momentum to the particle locations can in fact slow down convergence, and therefore use plain gradient descent with a decaying step size.

To trace an R-D upper bound for a given source, we solve the unconstrained R-D problem 7.3 for a heuristically-chosen grid of λ values similarly to those in [Yang and Mandt, 2022], e.g., on a log-linear grid $\{1e4, 3e3, 1e3, 3e2, 1e2, ...\}$. Alternatively, a constrained optimization method like the Modified Differential Multiplier Method (MDMM) [Platt and Barr, 1988] can be adopted to directly target $R(D)$ for various values of Ds. The latter approach will also

help us identify when we run into the $\log(n)$ rate limit of particle-based methods (Sec. 7.4.4): suppose we perform constrained optimization with a distortion threshold of D; if the chosen n is not large enough, i.e., $\log(n) < R(D)$, then no ν supported on (at most) n points is feasible for the given constraint (otherwise we have a contradiction). When this happens, the Lagrange multiplier associated with the infeasiable constraint (λ in our case) will be observed to increase indefinitely rather than stabalize to a finite value under a method like MDMM [Platt and Barr, 1988].

D.5.1 Deconvolution

Architectures for the neural baselines For the RD-VAE, we used a similar architecture as the one used on the banana-shaped source in [Yang and Mandt, 2022], consisting of two-layer MLPs for the encoder and decoder networks, and Masked Autoregressive Flow [Papamakarios et al., 2017] for the variational prior. For NERD, we follow similar architecture settings as [Lei et al., 2023a], using a two-layer MLP for the decoder network. Following Yang and Mandt [2022], we initially used softplus activation for the MLP, but found it made the optimization difficult; therefore we switched to ReLU activation which gave much better results. We conjecture that the softplus activation led to overly smooth mappings, which had difficulty matching the optimal $\nu^* = \alpha$ measure which is concentrated on the unit circle and does not have a Lebesgue density.

Experiment setup As discussed in Sec. 7.4.2, BA and our hybrid algorithms do not apply to the stochastic setting; to be able to include them in our comparison, the input to all the algorithms is an empirical measure μ^m (training distribution) with $m = 100000$ samples, given the same fixed seed. We found the number of training samples is sufficiently large that the losses do not differ significantly on the training distribution v.s. random/held-out samples.

Recall from Sec. 7.2.3, given data observations following $\mu = \alpha * \mathcal{N}(0, \sigma^2)$, if we perform deconvolution with an assumed noise variance $\frac{1}{\lambda}$ for some $\frac{1}{\lambda} \leq \sigma^2$, then the optimal solution to the problem is given by $\nu^* = \alpha_\lambda = \alpha * \mathcal{N}(0, \frac{1}{\lambda})$. We compute the optimal loss $OPT = \mathcal{L}_{BA}(\nu^*)$ by a Monte-Carlo estimate, using formula (D.5) with $m = 10^4$ samples from μ and $n = 10^6$ samples from ν^*. The resulting $\hat{\mathcal{L}}_{BA}$ is positively biased (it overestimates the OPT in expectation) due to the bias of the plug-in estimator for $\varphi(x)$ (D.4), so we use a large n to reduce this bias. [2] Note the same issue occurs in the NERD algorithm (7.13).

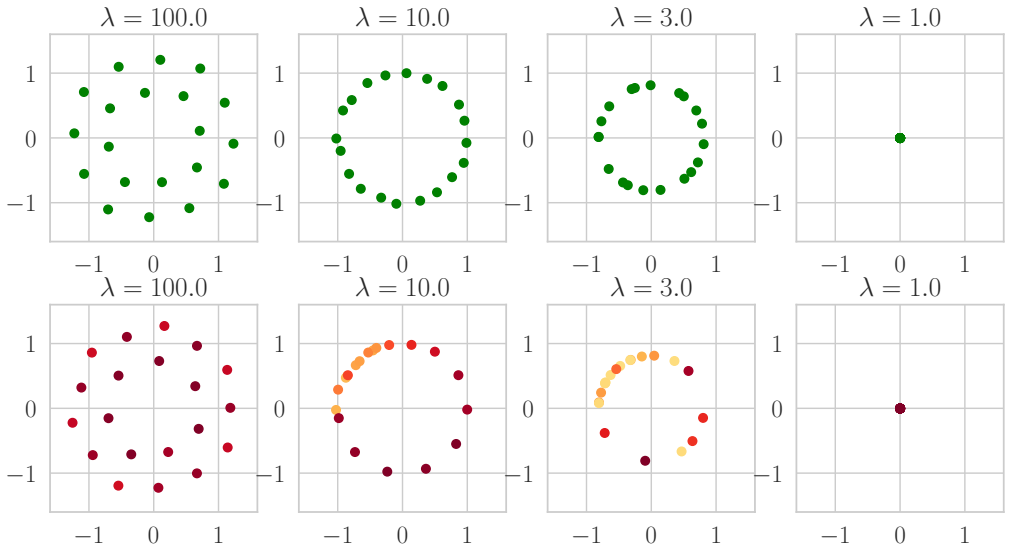


Figure D.1: Visualizing the converged $n = 20$ particles of WGD (top row) and hybrid WGD (bottom row) estimated on $m = 10000$ source samples in the deconvolution problem (Sec. 7.5.1), for decreasing distortion penalty λ. The case $\lambda = 10.0 = \sigma^{-2}$ corresponds to "complete noise removal" of the source and recovers the ground truth α (unit circle).

Optimized reproduction distributions To visualize the (continuous) ν learned by RD-VAE and NERD, we fit a kernel density estimator on 10000 ν samples drawn from each, using `seaborn.kdeplot`.

In Figure D.1, we provide additional visualization for the particles estimated from the training samples by WGD and its hybrid variant, for a decreasing distortion penalty λ (equivalently,

[2] Another, more sophisticated solution would be annealed importance sampling [Grosse et al., 2015] or a related method developed for estimating marginal likelihoods and partition functions.

increasing entropic regularization strength ϵ).

D.5.2 Higher-dimensional datasets

For NERD, we set n to the default 40000 in the code provided by [Lei et al., 2023a], on all three datasets.

For WGD, we used $n = 4000$ on *physics*, 200000 on *speech*, and 40000 on MNIST (see also smaller n for comparison in Fig. 7.5).

On *speech*, both NERD and WGD encountered the issue of a $\log(n)$ upper bound on the rate estimate as described in Sec. 7.4.4. Therefore, we increased n to 200000 for WGD and obtain a tighter R-D upper bound than NERD, particularly in the low-distortion regime. Such a large n incurred out-of-memory error for NERD.

We borrow the R-D upper and lower bound results of [Yang and Mandt, 2022] from their paper on *physics* and *speech*. We obtain sandwich bounds on MNIST using a similar neural network architecture and other hyperparameter choices as in the GAN-generated image experiments of [Yang and Mandt, 2022] (using a ResNet-VAE for the upper bound and a CNN for the lower bound), except we set a larger $k = 10000$ in the lower bound experiment; we suspect the resulting lower bound is still far from being tight.

D.6 Example implementation of WGD

We provide a self-contained minimal implementation of Wasserstein gradient descent on $\mathcal{L}_{BA}$, using the Jax library [Bradbury et al., 2018]. To compute the Wasserstein gradient, we evaluate the first variation of the rate functional in `compute_psi_sum` according to (D.6), yielding $\sum_{i=1}^{n} \psi^{\nu}(x_i)$, then simply take its gradient w.r.t. the particle locations $x_{1,\dots n}$ using

Jax's autodiff tool on line 51.

The implementation of WGD on $\mathcal{L}_{EOT}$ is similar, except the first variation is computed using Sinkhorn's algorithm. Both versions can be easily just-in-time (JIT) compiled and enjoy GPU acceleration.

```
# Wasserstein GD on the rate functional L_{BA}.
import jax.numpy as jnp
import jax
from jax.scipy.special import logsumexp

# Define the distortion function \rho.
squared_diff = lambda x, y: jnp.sum((x - y) ** 2)
pairwise_distortion_fn = jax.vmap(jax.vmap(squared_diff, (None, 0)), (0, None))


def wgrad(mu_x, mu_w, nu_x, nu_w, rd_lambda):
  """
  Compute the Wasserstein gradient of the rate functional, which we will use
  to move the \nu particles.
  :param mu_x: locations of \mu atoms.
  :param mu_w: weights  of \mu atoms.
  :param nu_x: locations of \nu atoms.
  :param nu_w: weights  of \nu atoms.
  :param rd_lambda: R-D tradeoff hyperparameter.
  :return:
  """

  def compute_psi_sum(nu_x):
    """
    Here we compute a surrogate loss based on the first variation \psi, which
    allows jax autodiff to compute the desired Wasserstein gradient.
    :param nu_x:
    :return: psi_sum = \sum_i \psi(nu_x[i])
    """
    C = pairwise_distortion_fn(mu_x, nu_x)
    scaled_C = rd_lambda * C  # [m, n]
    log_nu_w = jnp.log(nu_w)  # [1, n]

    # Solve BA inner problem with a fixed nu.
    phi = - logsumexp(-scaled_C + log_nu_w, axis=1, keepdims=True)  # [m, 1]
    loss = jnp.sum(mu_w * phi)  # Evaluate the rate functional. Eq (6) in paper.

    # Let's also report rate and distortion estimates (discussed in Sec. 4.4 of the paper).
    # Find \pi^* via \phi
```

```
    pi = jnp.exp(phi - scaled_C) * jnp.outer(mu_w, nu_w)  # [m, n]
    distortion = jnp.sum(pi * C)
    rate = loss - rd_lambda * distortion

    # Now evaluate \psi on the atoms of \nu.
    phi = jax.lax.stop_gradient(phi)
    psi = - jnp.sum(jnp.exp(jax.lax.stop_gradient(phi) - scaled_C) * mu_w, axis=0)
    psi_sum = jnp.sum(psi)  # For computing gradient w.r.t. each nu_x atom.
    return psi_sum, (loss, rate, distortion)

  # Evaluate the Wasserstein gradient, i.e., \nabla \psi, on nu_x.
  psi_prime, loss = jax.grad(compute_psi_sum, has_aux=True)(nu_x)
  return psi_prime, loss


def wgd(X, n, rd_lambda, num_steps, lr, rng):
  """
  A basic demo of Wasserstein gradient descent on a discrete distribution.
  :param X: a 2D array [N, d] of data points defining the source \mu.
  :param n: the number of particles to use for \nu.
  :param rd_lambda: R-D tradeoff hyperparameter.
  :param num_steps: total number of gradient updates.
  :param lr:  step size.
  :param rng: jax random key.
  :return: (nu_x, nu_w), the locations and weights of the final \nu.
  """
  # Set up the source measure \mu.
  m = jnp.size(X, 0)
  mu_x = X
  mu_w = 1 / m * jnp.ones((m, 1))
  # Initialize \nu atoms using random training samples.
  rand_idx = jax.random.permutation(rng, m)[:n]
  nu_x = X[rand_idx]  # Locations of \nu atoms.
  nu_w = 1 / n * jnp.ones((1, n))  # Uniform weights.
  for step in range(num_steps):
    psi_prime, (loss, rate, distortion) = wgrad(mu_x, mu_w, nu_x, nu_w, rd_lambda)
    nu_x -= lr * psi_prime
    print(f'step={step}, loss={loss:.4g}, rate={rate:.4g}, distortion={distortion:.4g}')

  return nu_x, nu_w


if __name__ == '__main__':
  # Run a toy example on 2D Gaussian samples.
  rng = jax.random.PRNGKey(0)
  X = jax.random.normal(rng, [10, 2])
  nu_x, nu_w = wgd(X, n=4, rd_lambda=2., num_steps=100, lr=0.1, rng=rng)
```