Interpolating Nonlinear Transforms to Parameterize an Image Compression Rate-Distortion Curve

David W. Dye
Department of Applied Mathematics
Harvard University
Cambridge, MA, USA
Email: ddye@college.harvard.edu

***Abstract*—Machine learning approaches to video and image compression face a fundamental problem in their ability to compress at variable bitrates and distortions. Current techniques for end-to-end visual media compression rely on the loss function to encode the tradeoff between compression and reconstruction quality, but in practice, it is often required that a model produces images at a desired bitrate automatically. In this case, current methods must train and deploy a new model to achieve the desired performance. Instead, we propose an interpolation solution allowing variable bitrate-achieving models to be computed on the fly and accessible to both encoding and decoding parties. We investigate the performance of our model by comparing the position of interpolated models to the position of trained models in rate-distortion space, taking trained models to be the ground truth for models lying on the rate-distortion curve. The MSE between the input and output images from a model was used to measure distortion during training, and the multi-scale structural similarity index measure (MS-SSIM) was used to measure distortion during model evaluation. Both training and testing was done on disjoint subsets of the Common Objects in Context (COCO) image database. We observed no qualitative difference in models trained on interpolated points versus those trained on the loss function for a desired bitrate and distortion; as a direct result, we claim that the rate-distortion curve can be closely approximated and fully parameterized by interpolated models.**

I. Introduction

In today's digital environment, there is an ever-increasing need to transmit information quickly in large quantities. Of all forms of information available on the internet, video is by far the largest: digital media streaming comprises over half of the total internet traffic today [1]. Any method that compresses video files into a smaller bit representation without losing quality could have an enormous impact on the total amount of space and time needed to store and transmit those files in bulk.

Video compression is built on top of image compression. Standard video compression algorithms such as H.264 use I-frames to periodically encode the full frame information of the video, and they fill in the gaps between I-frames with P or B-frames that contain only the frame-by-frame changes in pixel intensities [2]. Thus, improving video compression can be decomposed into improving compression for intraframe (spatial) data in I-frames and improving compression for intraframe and interframe (temporal) data in P and B-frames.

Recent research has moved towards the goal of image and video compression through end-to-end machine learning (ML) algorithms [3], [4]; that is, feed an image or video file into a pre-trained ML encoder to produce a compressed string of bits, then use a pre-trained ML decoder to convert this string into a lossy representation of the original image. Various methods have been used for compression: autoencoders (AEs), variational AEs, convolutional neural networks (CNNs), recurrent neural networks (RNNs), and generative adversarial networks (GANs) have all been surveyed and compared against one another [5]–[7].

In this paper, we consider models trained with the purpose of achieving a rate-distortion tradeoff, where this tradeoff is encoded in the loss function of the model. These models aim to find a feature representation for images through some nonlinear transform and its inverse, where the feature representation can be efficiently compressed and transmitted using a lossless coding scheme. Pioneering research in this area was done by Ballé *et al.* in [3] and [8]; here, we extend the methods and architectures Ballé *et al.* developed to further characterize models lying on the rate-distortion curve resulting from such loss functions.

The goal of this paper is to address the problem of variable bitrate encoding in end-to-end image compression. This arises in applications where a desired bitrate or distortion is required, and it has been studied extensively for video compression [9], [10]. We propose and analyze an interpolation-based framework for variable-bitrate end-to-end image compression. Intuitively, our model tracks a minima in the loss function as it changes as a function of a rate-distortion tradeoff parameter. Ultimately, we show that our interpolation-based approach achieves rate-distortion performance qualitatively identical to that of models trained on the loss function directly.

II. Notation

Let C_x and C_y be the number of channels in an input image and its feature representation, respectively, and let $\mathcal{X} \subset \mathbb{R}^{C_x \times N_x \times M_x}$ and $\mathcal{Y} \subset \mathbb{R}^{C_y \times N_y \times M_y}$ be the set of possible images and their feature representations. Let $\lambda \in (0, 1)$ be a rate-distortion tradeoff parameter, let $\boldsymbol{\theta}$ parameterize some nonlinear transform $f(x; \boldsymbol{\theta})$ mapping an image $x \in \mathcal{X}$ to a compressible feature representation $y \in \mathcal{Y}$, and let $\boldsymbol{\theta}(\lambda)$ be the parametrization for the nonlinear transform $f(x; \boldsymbol{\theta}(\lambda))$

lying on the rate-distortion curve for a given λ. Then, let $q : \mathcal{Y} \to \mathcal{Q} \subset \mathbb{Z}^{C_y \times N_y \times M_y}$ be a quantization function taken to be round$(\cdot)$ in this work, and let $g(\cdot; \boldsymbol{\phi}) : \mathcal{Q} \to \hat{\mathcal{X}}$ be a learned inverse transform to $f(\cdot; \boldsymbol{\theta})$ after quantization. We use a hat, $\hat{\cdot}$, to represent reconstructed quantities. Let $P_{\boldsymbol{q}}(\cdot; \boldsymbol{\rho})$ represent the learned joint probability distribution for quantized feature space, and let $d(x, \hat{x}) : (\mathcal{X}, \hat{\mathcal{X}}) \to \mathbb{R}$ be a distortion measure between x and $\hat{x}$, taken to be mean squared error (MSE) in this work.

III. Methods

The rate-distortion curve for a given compression problem represents the boundary of a convex set of possible solutions satisfying compression at a certain rate and maximum distortion. It is natural to think of rate and distortion as a tradeoff: compression can be improved at the expense of an increase in distortion. The goal of this section is to parameterize the rate-distortion curve for image compression with a set of learned nonlinear transforms converting images to a more easily-compressible feature space.

A. Network Architectures

Because the rate-distortion curve is one-dimensional, it is sensible to define models as functions of a "rate-distortion tradeoff" parameter, $\lambda \in (0, 1)$, that parametrizes this curve. We impose that $\lambda \to 0$ represents models only maximizing rate and $\lambda \to 1$ represents models only minimizing distortion, giving a loss function enforcing the rate-distortion tradeoff as:

$$\begin{aligned}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \lambda, x) = -(1-\lambda)\mathbb{E}[\log_2 P_{\boldsymbol{q}}(q(f(x; \boldsymbol{\theta})); \boldsymbol{\rho})] \\ + \lambda \mathbb{E}[d(x, g(q(f(x; \boldsymbol{\theta})); \boldsymbol{\phi})], \quad (1)\end{aligned}$$

where this loss function is equivalent under optimization to that in [3]. For a given choice of λ, minimizing (1) gives parameters $\boldsymbol{\theta}(\lambda)$, $\boldsymbol{\phi}(\lambda)$, and $\boldsymbol{\rho}(\lambda)$, which together fully determine models lying on the rate-distortion curve.

For clarity, we explain the optimization process for fixed λ. First, an image is sampled as $x \in \mathcal{X}$. Then, the nonlinear transform is applied: $y = f(x; \boldsymbol{\theta}) \in \mathcal{Y}$. During learning, applying q would kill gradients, so instead distortion is achieved by simulating quantization through the addition of uniform noise: $q_i^{\text{learning}}(y_i) = y_i + Z$, where $Z \sim \text{Unif}(-0.5, 0.5)$, and i indexes over all channels, rows, and columns of $\mathcal{Y}$. We then perform the inverse nonlinear transform $g(q(y); \boldsymbol{\theta})$ and measure distortion of the reconstruction from the original image using $d(x, \hat{x})$.

We follow the network implementation in [3] to learn the nonlinear transforms f and g. For each network layer in f, we perform a strided convolution and a generalized divisive normalization (GDN) nonlinearity to normalize and whiten the input with learnable parameters, and for each network layer in g, we perform a fractional-strided (transposed) convolution with an inverse GDN nonlinearity (see Fig. 1). More details on the network architecture may be found in [3].

To compute $P_{\boldsymbol{q}}$, we simultaneously optimize $\boldsymbol{\rho}$ with $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$; that is, we couple two optimization steps together.

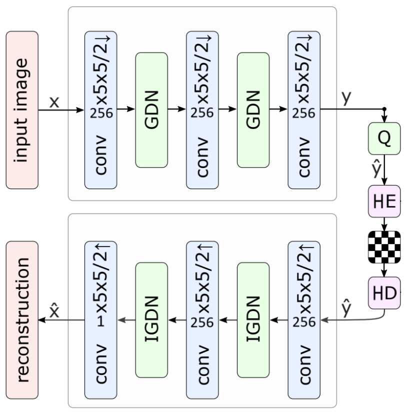


Fig. 1. Model architecture, adapted from [3] and [8]. Instead of 128 or 196 channels, our model uses 256 channels, and instead of arithmetic coding, our model uses Huffman coding.

We assume that the feature representation has independent elements, giving:

$$P_{\boldsymbol{q}}(q(y); \boldsymbol{\rho}) = \prod_{i=1}^{C_y N_y M_y} P_{q_i}(q(y_i); \boldsymbol{\rho}_i). \quad (2)$$

This is a poor assumption in practice, but it is enough to provide a basic entropy estimate for the loss function, and weakening it requires much more sophisticated network architectures [3], [8], [11]. We follow the framework developed in [8] to learn $\boldsymbol{\rho}_i$. For each $i \in \{1, \ldots, C_y \times N_y \times M_y\}$, we have a neural network learning the probability distribution for that feature space pixel. This is implemented in the code as a network of two hyperbolic tangent activation layers and a final sigmoid layer to convert to a probability. To improve computational efficiency, each network is optimized simultaneously using standard stochastic gradient descent (SGD). The loss function in this case is given by:

$$\mathcal{L}_{\boldsymbol{\rho}}(\boldsymbol{\rho}; q(y)) = -\sum_{i=1}^{C_y N_y M_y} \log\left[P_{q_i}(q(y_i); \boldsymbol{\rho}_i)\right]. \quad (3)$$

More details and mathematical formalism for probability density estimation neural networks may be found in Appendix (6.1) of [8].

B. Encoding and Decoding

Compression rate is dependent on the quality of the learned distribution $P_{\boldsymbol{q}}$ and the ability to translate a quantized (integer-valued) feature tensor q to a binary string that can be entropy coded. Ballé *et al.* [3] recommends integer arithmetic encoding followed by CABAC entropy encoding, similar to the H.264 and HEVC video coding standards [12], but we instead performed Huffman coding using $P_{\boldsymbol{q}}$, followed by ZIP-style compression using zlib's Deflate lossless compression algorithm. This compression scheme is slower and may encode an insufficient number of integers due to practical limitations. Nonetheless, it provides a baseline estimate for bitrate (bits per pixel) that can be compared between models. Compression performance can be significantly improved under different model architectures, such as those acounting for context-dependence in the feature layer [8], [11], but that is not within the scope of this paper.

However, it is unacceptable to have a codebook that cannot encode all possible symbols; for a given image, we expect to see several improbable symbols. To account for this, we use exponential-Golomb coding [12] to encode integers that appear with low probability. This is determined in practice by integers far from 0, such as $|q_i| > 50$, which is improbable due to batch normalization.

Note that encoding and decoding are both lossless using either the Huffman and Deflate algorithm or the CABAC algorithm; all information loss in the model comes entirely from the interplay between the rate and distortion terms of (1).

C. Model Interpolation

We now consider the problem of querying models at a user-specified λ^*. This arises in applications where a user wishes to encode an image at a target bitrate or distortion, but only a small set of pretrained models are available. It is possible that no model in the pretrained set is suitable for the user's application, and naively interpolating model weights from this set to obtain a new model will also fail due to invariances in the convolutional neural network. Different training initializations lead to vastly different models with similar performance, so there is no guarantee that constructing a model with λ^* from known models with $\lambda_1 < \lambda^* < \lambda_2$ will give good performance.

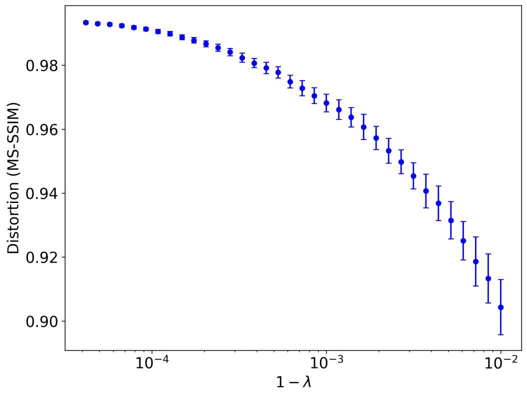
Intuitively, an "interpolation" solution to (1) is not necessarily given by the global minimum of the loss function for a given λ, but rather the local minimum that is "closest" to the minimum found by minimizing (1) for some other nearby λ that is treated as an interpolation point. This is to enforce locality: the best guess for how the loss function minimum evolved with λ is made by finding the minimum nearest where it was before. Let λ_1 be a choice of λ for which (1) and (3) have been solved; that is, we have access to $\boldsymbol{\theta}(\lambda_1)$, $\boldsymbol{\phi}(\lambda_1)$, and $\boldsymbol{\rho}(\lambda_1)$. Then, the interpolation solution for a model parameterized by λ_2 is given by:

$$\boldsymbol{\theta}(\lambda_2), \boldsymbol{\phi}(\lambda_2), \boldsymbol{\rho}(\lambda_2) = \arg\min_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\rho}}\{\mathcal{D}_{\lambda_1}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\rho})\}, \tag{4}$$

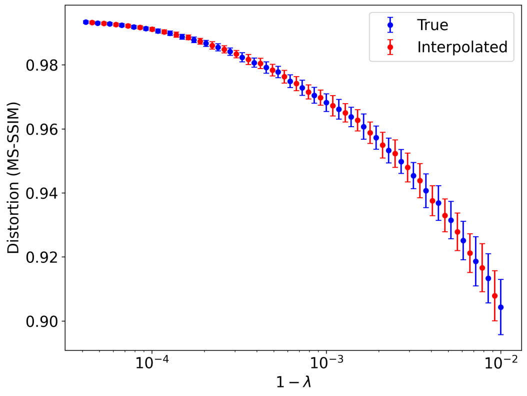
where $\mathcal{D}_{\lambda_1}$ is the set of distances between weights at each minima:

$$\mathcal{D}_{\lambda_1}(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\rho}) = \Big\{\|\boldsymbol{\theta} - \boldsymbol{\theta}(\lambda_1)\|_2^2 + \|\boldsymbol{\phi} - \boldsymbol{\phi}(\lambda_1)\|_2^2 + \|\boldsymbol{\rho} - \boldsymbol{\rho}(\lambda_1)\|_2^2 : \nabla\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \lambda_2) = \mathbf{0}, \nabla\mathcal{L}_{\boldsymbol{\rho}}(\boldsymbol{\rho}) = \mathbf{0}\Big\}. \tag{5}$$

Finding all elements of this set, which may even be infinite, is impractical from a computational perspective. However, it does suggest a practical approach: use λ_1 model weights as initial values for SGD on λ_2 model loss functions. Then, if no minima-destroying bifurcation occurs and $\|\lambda_1 - \lambda_2\|_2^2 < \varepsilon$ for some ε sufficiently small, the solution to (4) will be the result of SGD on (1) and (3).



(a) Distortions from fitted models in the set Λ.



(b) Distortions from fitted models in the set Λ with interpolated results.

Fig. 2. Average MS-SSIM values of various models as functions of $1 - \lambda$. As $\lambda \to 1$, the MS-SSIM goes to 1 as well, indicating almost perfect reconstruction. The average and standard deviations are computed over the 100 test images.

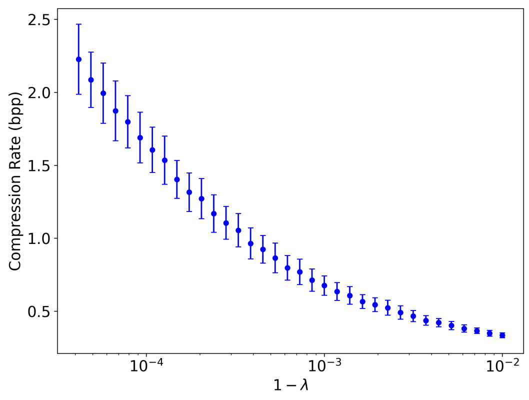
There are an infinite number of possible interpolation methods, but for simplicity we only consider linear interpolation between consecutive interpolation points. The linear interpolation weight function is:

$$w_k(\lambda) = \frac{\lambda_{k+1} - \lambda}{\lambda_{k+1} - \lambda_k}. \tag{6}$$

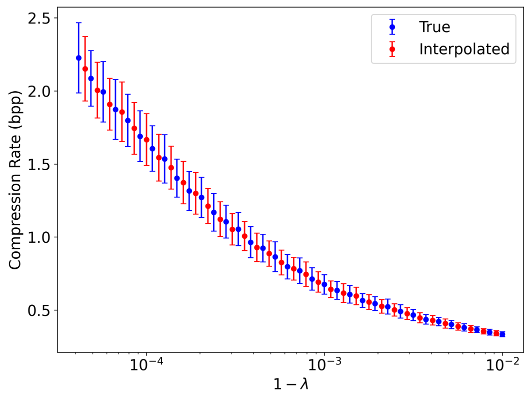
Then, the process for creating an interpolation model is as follows:

1) Choose L values of $\lambda \in (0, 1)$ and create an ordered set $\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_N\} \in (0, 1)^N$, $\lambda_1 > \lambda_2 > \cdots > \lambda_N$.
2) Train an initial model for λ_1 by minimizing (1) and (3).
3) For $k = 2, \dots, N$, use parameters for model $k - 1$ as initial parameters for model k. Then train model k using SGD.
4) Given a valid query for λ^*, find k such that $\lambda_k < \lambda^* < \lambda_{k+1}$. Then, $\boldsymbol{\theta}(\lambda^*)$, $\boldsymbol{\phi}(\lambda^*)$, and $\boldsymbol{\rho}(\lambda^*)$ can be found by interpolation. For example, $\boldsymbol{\theta}(\lambda^*)$ is given by:

$$\boldsymbol{\theta}(\lambda^*) = w_k(\lambda^*)\boldsymbol{\theta}(\lambda_k) + (1 - w_k(\lambda^*))\boldsymbol{\theta}(\lambda_{k+1}). \tag{7}$$

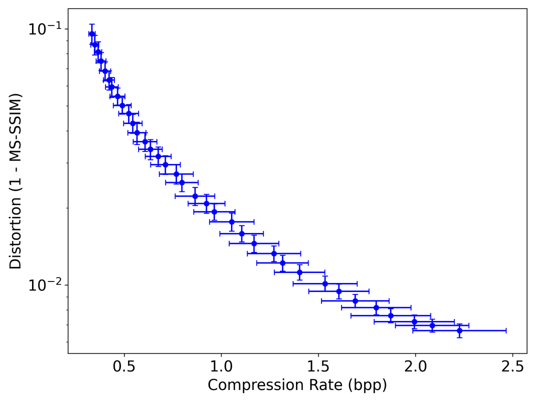


(a) Rates from fitted models in the set Λ.

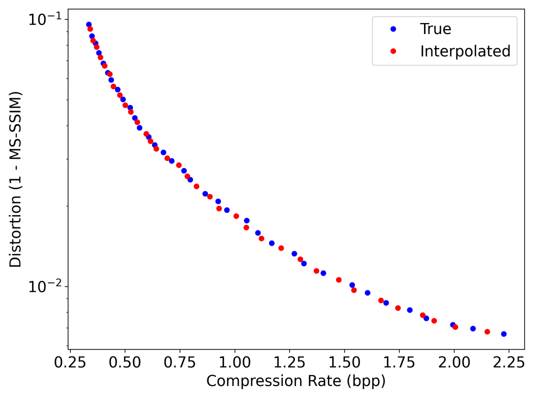


(b) Rates from fitted models in the set Λ with interpolated results.

Fig. 3. Average compression rates of various models as functions of $1 - \lambda$. As $\lambda \to 1$, the number of bits per pixel required for compression increases. The average and standard deviations are computed over the 100 test images.



(a) Rate-distortion curve for fitted models in the set Λ.



(b) Rate-distortion curve with interpolated models.

Fig. 4. Computed rate-distortion curves are approximately convex. Interpolated points have excellent performance, indistinguishable from fitted rate-distortion points. The average and standard deviations are computed over the 100 test images.

This creates a framework for quickly approximating model parameters on untrained values of λ lying within the interpolation range.

IV. Results

We trained and evaluated model performance on a subset of 10000 images from Microsoft's Common Objects in Context (COCO) database [13] using the multi-scale structural similarity index measure (MS-SSIM) with hyperparameters tuned for human perception as in [14]. Although MS-SSIM was used to compare models, mean squared error (MSE) was used for the distortion function during model training as it is more robust and easier to compute.

We used a three-layer model architecture for the encoder and decoder with the same setup as in [3] and [8], except with 256 channels in the convolutional layers instead of 128. For simplicity, we grayscaled all images, removed any image smaller than 256×256, removed additional images at random to end up with 8196 total, and sampled random 256×256 subimages when training the model. The remaining 8196 images were split into a training set of 8096 images and a testing set of 100 images to evaluate performance. The model is not restricted to 256×256 images for nonlinear transformations, but the entropy coding stage requires inputs to match the training size since a probability distribution is learned for each channel in the feature space, and the number of channels depends on the model input's dimensions.

Models were trained using an NVIDIA GeForce RTX 4090 GPU rented online through RunPod, an AI training service that provides access to powerful GPUs. We trained 35 models in total, with the first trained for 140 epochs at a learning rate of 0.001, and the following models trained for 20 epochs also at a learning rate of 0.001. Fewer training epochs were required during interpolation phases since the model was already close to equilibrium, and each value of λ_k created only a small perturbation to the loss function and its minimum. We used $\lambda_k = 1 - 10^{-4.3793+k/14.23}$, $k = \{0, \ldots, 34\}$ as the interpolation points, giving an interpolation range of $\lambda^* \in [0.99, 0.99996]$. Distortion increases rapidly for decreasing λ (see Fig. 2), whereas compression bitrate increases rapidly for increasing λ (see Fig. 3).

To evaluate whether the interpolation framework given in Section III-C is reasonable for nonlinear transform coding, we chose values of λ located at the arithmetic mean of consecutive

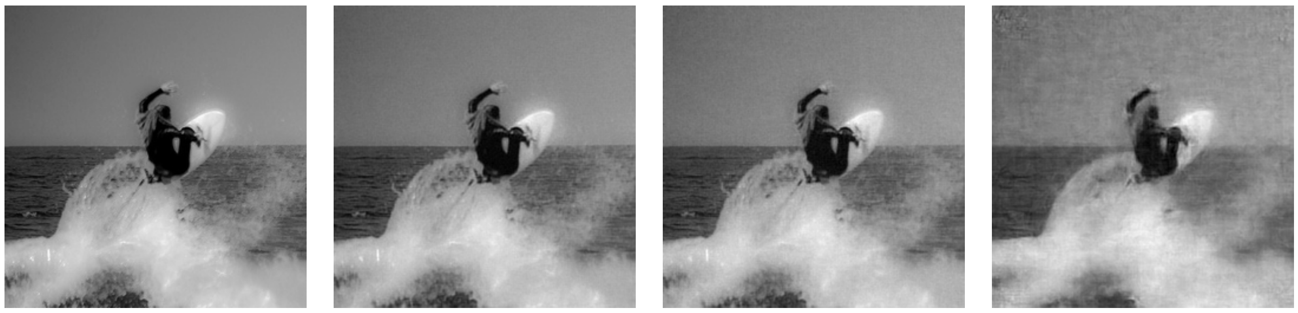

(a) Original (b) MS-SSIM: 0.994, 2.19 bpp (c) MS-SSIM: 0.988, 1.22 bpp (d) MS-SSIM: 0.941, 0.39 bpp

Fig. 5. Compression of “Surfer” image at increasing distortions and decreasing bitrates. The values of λ are 0.99996, 0.99976, and 0.99389 from (b) to (d).

interpolation points and measured performance via the bitrate, MS-SSIM, and their sample standard deviations. The rate-distortion curves both with and without these midpoints are shown in Fig. 4.

Finally, we show an example test image compressed using models trained at different values of λ in Fig. 5. The highest λ model reconstructs the test image almost perfectly since a high bitrate is allowed, but the rate-distortion tradeoff is demonstrated as decreasing bitrate leads to qualitatively worse reconstructions.

V. Discussion

To a large degree, this work is an extension of the results and architectures presented by [3] and [8], but there are several important distinctions that lead to different and novel results. In this section, we discuss these differences and areas for improvement.

First, neither [3] nor [8] discusses choices of λ or the functional relationship between λ, bitrate, and distortion. The results presented in Fig. 2 and Fig. 3 explain this relationship graphically. In practice, it is crucial to prioritize the distortion term in the loss function by choosing $\lambda \geq 0.99$ in order to avoid useless reconstructions.

Most importantly, we have demonstrated that it is possible to construct models performing as well as those obtained by minimizing (1) and (3) without explicitly performing the minimization. By pretraining a sequence of models according to the interpolation method described in Section III-C, it is possible to quickly find other models lying on the rate-distortion curve for this nonlinear transform model architecture. A more in-depth analysis of this behavior is a natural extension of this work; for instance, determining the largest gap between λ_k and λ_{k+1} for which interpolation still produces reasonable models. If this gap is large, the number of stored models required to parameterize the entire rate-distortion curve will be small.

There are several limitations to the analysis performed here. First, the utility of nonlinear transform models is still questionable, since applying the transform and encoding is strictly slower than traditional discrete cosine transform (DCT)-based linear transforms. This makes the application of such models to problems like video compression and livestreaming currently implausible. Additionally, the amount of space required to store multiple transform models for interpolation may be intractable for some applications. This problem worsens as models become more complicated, as is the case in [8] and [11], which attempt to capture context information through additional model layers to improve entropy coding.

Despite these limitations, the ability to interpolate along the rate-distortion curve is an important property that enables variable-bitrate image coding and warrants additional research. It is possible that alternative methods for fitting model parameters given a set of known, fitted parameters rather than pure interpolation may give even better results, potentially further reducing the number of required interpolation points.

VI. Conclusion

We have shown that it is possible to parameterize the rate-distortion curve by interpolating between a small set of fixed models. To do so, we first recreated results from Ballé *et al.* [3] by developing a codebase for image compression through nonlinear transform coding. Fitting a sequence of models using the parameters for one model as the initial condition for the next allowed us to track a local optimum of the loss functions as they shifted from minimizing distortion to minimizing rate. The techniques used here can easily be extended to color images and alternative interpolation methods, and further research is required to understand the interplay between model complexity and maximum interpolation distance. By numerically demonstrating that interpolated models have the capacity to perform as well as those trained on a loss function directly, we have achieved a variable bitrate encoding scheme for end-to-end image compression.

VII. Acknowledgments

I would like to thank Professor Calmon and Dr. Vithana for putting together this excellent course on information theory. The course and this project have provided outstanding preparation for using information theory in my research.

References

[1] O. Rippel, L. Bourdev, D. Precup, and Y. Teh, “Real-time adaptive image compression,” in *International Conference on Machine Learning, Vol 70*, vol. 70, (San Diego), Journal of Machine Learning Research, 2017.

[2] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the h.264/avc video coding standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[3] J. Ballé, V. Laparra, and E. P. Simoncelli, “End-to-end Optimized Image Compression,” in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[4] E. Agustsson, D. Minnen, N. Johnston, J. Balle, S. J. Hwang, and G. Toderici, “Scale-Space Flow for End-to-End Optimized Video Compression,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (Los Alamitos), pp. 8500–8509, IEEE, IEEE, 2020.

[5] S. Jamil, M. J. Piran, M. Rahman, and O.-J. Kwon, “Learning-driven lossy image compression: A comprehensive survey,” *Engineering applications of artificial intelligence*, vol. 123, p. 106361, 2023.

[6] H. Liao and Y. Li, “LFC-UNet: learned lossless medical image fast compression with U-Net,” *PeerJ. Computer science*, vol. 10, pp. e1924–e1924, 2024.

[7] F. Mentzer, E. Agustsson, J. Ballé, D. Minnen, N. Johnston, and G. Toderici, “Neural Video Compression Using GANs for Detail Synthesis and Propagation,” in *ECCV 2022*, vol. 13686 of *Lecture Notes in Computer Science*, (Switzerland), pp. 562–578, Springer, 2022.

[8] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, “Variational image compression with a scale hyperprior,” *arXiv (Cornell University)*, 2018.

[9] P. Pancha and M. El Zarki, “Mpeg coding for variable bit rate video transmission,” *IEEE Communications Magazine*, vol. 32, no. 5, pp. 54–66, 1994.

[10] M. Yang, X. Mao, Y. Yin, Z. Zhu, D. Wang, S. Wan, and F. Yang, “Learning-based video compression with continuously variable bitrate coding,” in *2024 IEEE International Conference on Image Processing (ICIP)*, pp. 3723–3729, 2024.

[11] Y. Hu, W. Yang, Z. Ma, and J. Liu, “Learning end-to-end lossy image compression: A benchmark,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 8, pp. 4194–4211, 2022.

[12] I. E. G. Richardson, *The H.264 advanced video compression standard*, ch. 7. Chichester, West Sussex, UK: Wiley, 2nd ed. ed., 2010.

[13] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, T. Tuytelaars, D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars, B. Schiele, T. Pajdla, and D. Fleet, “Microsoft coco: Common objects in context,” in *Computer Vision - ECCV 2014, Pt V*, vol. 8693 of *Lecture Notes in Computer Science*, (Cham), pp. 740–755, Springer International Publishing, 2014.

[14] Z. Wang, E. Simoncelli, A. Bovik, and M. Matthews, “Multi-scale structural similarity for image quality assessment,” in *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems & Computers, Vols 1 and 2*, (New York), pp. 1398–1402, IEEE, 2003.