Learned Image Compression with Perceptual Quality at Matched Bitrates

Sai Dore
Duke University
Durham, NC 27708
sai.dore@duke.edu

Supawich Puengdang
Duke University
Durham, NC 27708
supawich.puengdang@duke.edu

Jinze Lyu
Duke University
Durham, NC 27708
jinze.lyu@duke.edu

Abstract

This project investigates how a compact learned compression model balances bitrate, distortion, and perceptual quality. The model uses a lightweight convolutional autoencoder with a learned entropy model, and several experiments study how different loss terms and priors affect performance. Starting with an MSE-only rate–distortion setup, MS-SSIM and LPIPS perceptual losses are added and their impact is evaluated, showing that MS-SSIM consistently reduces quality while a small LPIPS weight improves perceptual metrics with only a minor bitrate increase. Comparing a factorized Gaussian prior to a simplified hyperprior shows that the factorized prior provides better compression and reconstruction quality in this small-model setting. The final system is also evaluated against JPEG and WebP on an external dataset, where it achieves better perceptual quality in the very low-bitrate regime, while traditional codecs remain stronger at higher bitrates. Overall, the study provides a clear picture of how perceptual losses and entropy models affect learned compression at low rates and identifies paths for future improvements.

1 Introduction

1.1 Background and Motivation

Because raw RGB image files are extremely large, practical systems rely on compression to stay within storage and bandwidth limits, cutting storage costs and improving delivery over fixed networks. Traditional image codecs such as JPEG and WebP address this need by using hand-designed pipelines, and they do this by applying fixed transforms, quantizing the transform coefficients, and then applying entropy coding. These codecs generally work well and are deployed everywhere, but they were engineered around simple distortion metrics like mean squared error (MSE) and its companion peak signal-to-noise ratio (PSNR), which do not always match perceived visual quality. As a result, interest in compression methods that are data-adaptive and more closely aligned with human perception has been growing. Learned image compression uses neural networks to replace the fixed transform with a learned encoder-decoder structure, and this can be trained to maximize both PSNR and other perceptual metrics. This project studies how far a compact learned compressor can push perceptual quality at low bitrates, and how its behavior compares to established codecs.

1.2 Learned Image Compression and Perceptual Metrics

Learned image compression replaces the hand-crafted transforms of the traditional codecs with a neural network trained end-to-end for compression. A convolutional encoder maps the input image to a lower-dimensional latent representation, and a decoder reconstructs an image from this latent. A separate entropy model (prior) is trained to predict how likely each latent value is, giving an estimate of

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

how many bits per pixel (bpp) would be required to transmit the compressed representation. Training uses a rate-distortion objective, where the model minimizes a weighted sum of an estimated rate term (bpp from the entropy model) and a distortion term that measures how different the reconstruction is from the original. MSE and PSNR are common choices for the distortion term, but they do not always accurately track human perception. To better capture how images actually look, other perceptual quality metrics have been produced. These include multi-scale structural similarity (MS-SSIM), which focuses on structural and contrast information, and Learned Perceptual Image Patch Similarity (LPIPS), which measures distances between feature vectors in a neural network. In this paper, these metrics are used both as evaluation tools and as additional loss terms on top of MSE.

1.3 Goals and Contributions

The overall goal of this project is to build and analyze a compact learned compressor model and to analyze how different loss terms and entropy models affect its rate-distortion perceptual tradeoff. The project is structured around a set of experiments on loss weights, entropy models, and codec comparisons to understand the rate–distortion–perception trade-offs of a compact model. The main contributions include a systematic empirical study of how MSE, MS-SSIM, and LPIPS affect a compact compressor, a comparison of factorized versus hyperprior entropy models within the same framework, and a practical benchmark of the best learned model against standard codecs. The methodology and results for each of these experiments are discussed thoroughly in following sections.

2 Related Works

Ballé et al. (2017) introduced a fully end-to-end optimized learned image compression system, based on a convolutional analysis-synthesis transform and a factorized Gaussian entropy model over quantized latents [1]. Their work identified compression as a differentiable rate-distortion optimization problem, and they used uniform noise to approximate quantization during training and estimated rate via a learned density model, with MSE and PSNR being the distortion term and quality metric respectively. This paper follows the same framework, but focuses on a smaller lightweight model and extends the study in a few directions Ballé et al. briefly expanded on: comparing different distortion and perceptual losses and their weights in the training objective, and comparing a simple factorized prior to a hyperprior using the same implementation.

Ballé et al. (2018) extended their previous framework by introducing a scale hyperprior, which adds a second latent space to model spatially varying uncertainty in the latents [2]. The hyperprior predicts per-location scales for the latent channels instead of assuming a fixed Gaussian prior, leading to better rate-distortion performance and showing that compression efficiency can be improved by learning this hierarchal prior. In this paper, the same framework is implemented on a smaller scale, where a simplified hyperprior variant is implemented in place of the factorized Gaussian prior, and the two are compared within the same lightweight autoencoder with the same training setup. The authors focus on PSNR and compression, while this implementation studies how perceptual losses interact with the entropy models and benchmarks the best configuration against codecs at matched bitrates.

3 Methodology

3.1 Model Architecture

The system's core is a convolutional autoencoder that maps RGB images to a compact latent representation and back. The encoder is a stack of convolutional layers with downsampling, which reduce the spatial resolution and increase the number of feature channels. Given an image x $\in [0,1]^{3\times H\times W}$, the encoder produces a latent tensor y $\in \mathbb{R}^{C\times H'\times W'}$, where C is the number of latent channels (128), and H' and W' are the reduced spatial dimensions. The decoder copies this structure via upsampling and convolutions and reconstructs an image from the latent representation. Together, the encoder and decoder form a lightweight model. To make latent representation compressible, a learned per-channel quantization scale (q_{scale}) is introduced with shape (C,1,1), where y is divided by this scale before quantization in Equation 1 below, so the network can learn an effective step size for each channel.

$$y_{\text{scaled}} = \frac{y}{q_{\text{scale}} + \epsilon} \tag{1}$$

During training, two parallel quantization paths are used. The soft path adds uniform noise in the range of [-0.5,0.5] to y_{scaled} to approximate quantization, while the hard path applies rounding with a straight-through estimator (STE) so that gradients can pass through. Both are then rescaled to the original latent domain by multiplying q_{scale} before decoding. The decoder is applied to both versions of the latent, creating soft and hard reconstructions, where the soft branch provides a differentiable signal during training and the hard branch matches the behavior of the rounded latents during testing. In the loss, the two reconstructions are combined with weights of 0.7 for soft and 0.3 for hard, so optimization can be guided by both. during evaluation, only the hard-quantized latents are used, so the measured quality and bitrate correspond to what would be seen in an actual deployed compressor.

3.2 Entropy Models and Rate Estimation

In this model, the entropy prior assigns probabilities to the latent tensor y, which is used to estimate the bits per pixel (bpp) needed under an ideal entropy coder. For most of the experiments, a factorized Gaussian prior is used: each element of y is modeled as an independent zero-mean Gaussian with a learned standard deviation of σ_c for each channel. During training, the scaled latents are quantized (soft or hard), then rescaled by q_{scale}. For the rate term, the soft quantized latent on the original scale is fed into the prior and a log-probability under the Gaussian is computed for each element. The estimated bitrate is then computed using the average negative-log-likelihood shown in Equation 2, with N being the batch size and H and W being the original image dimensions. This gives a continuous, idealized bpp, as it is assumed an ideal entropy coder with complete coding efficiency is being used and the Gaussian prior is a good fit for the latent distribution. These values are used to compare different loss settings within the same model, compare the different priors under the same rate estimate, and to see if the optimistic bpp can beat codec bpps.

$$R \approx \frac{1}{NHW} \sum_{n,c,h,w} -\log_2 p(y_{n,c,h,w}) \tag{2}$$

A simple hyperprior variant based on Ballé et al's implementation is also used in place of the factorized prior. A simple global σ_c per channel can be too limited, as some spatial locations can be harder to predict than others, so the hyperprior introduces a second latent tensor z that summarizes the variability in y. A small hyper-encoder transforms y into a lower-resolution z, which is quantized and passed through a hyper-decoder that predicts spatially varying scale parameters for the Gaussian prior over y, giving a conditional prior p(y|z). The total rate is then the sum of the bits to encode y under the conditional Gaussian prior and the bits to encode z under a simpler factorized prior. This usually increases bpp but can better match the latent distribution, and a simple version is used here to keep the model size close to the baseline and compare it to the factorized prior in the experiments.

3.3 Loss Function, Data, and Training Setup

3.3.1 Loss Function

The model is trained with a rate-distortion-perception loss shown in Equation 3, where R is the estimated bpp from the entropy model, MSE is the pixel-wise mean squared error, MS-SSIM is a structural similarity score in [0,1], and LPIPS is learned perceptual distance based on deep features.

$$L = \lambda R + \text{MSE} + \beta(1 - \text{MS-SSIM}) + \gamma\,\text{LPIPS} \tag{3}$$

The weights λ, β, and γ control which factors are weighed more in the loss. For experiment one, β = γ = 0, and λ is varied to produce pure MSE rate-distortion curves. For experiment two, λ is fixed to a value determined in experiment one, γ = 0, and β is varied to add MS-SSIM loss into the picture. For experiment three, γ is now varied to add LPIPS into the combined loss function. During training, soft and hard reconstructions are decoded, and their MSEs are combined with weight 0.7 for soft and 0.3 for hard, so the loss sees both the smooth and quantized paths.

3.3.2 Data and Training Setup

A set of high-resolution RGB images from Div2K is used and split into a training set (360 images) and a small validation set (40 images). These images are resized and cropped to 256x256 patches, and a random horizontal flip is applied if training, and the pixels are mapped to [0,1]. This setup

allows for training efficiency while also exposing the model to different structures and textures. For the JPEG/WebP comparisons and visual crops, an external test set from Kodak24 is used, consisting of 24 small natural images used for compression benchmarks. This allows for a fair comparison between the learned compression model and established codecs using a variety of metrics.

For experiment one, a cosine annealing learning rate starting at 3e-4 over 5000 steps is used, compared to the learning rate starting at 1e-4 and training lasting for 2000 steps for experiments two and three. For these experiments, the best checkpoint from the previous experiments are used for training, so the model can be fine-tuned. A batch size of 16 images per step and an image patch size of 256x256 is used for all experiments, along with the Adam optimizer, while training uses mixed precision on GPU to reduce memory and speed up training. This setup gives a reproducible model that can be trained in a few hours while still allowing for the exploration of how the rate-distortion-perceptual trade-offs can be affected with different losses and entropy models.

4 Experiments

All experiments use the same training and validation split, crop size, batch size, and cosine annealing learning rates as described in the previous section. For the learned model, the reported bpp comes from the entropy model, giving an optimistic lower bound of the number of bits under an ideal coder, whereas JPEG/WebP values are computed from file sizes and include overhead.

4.1 Experiment 1: MSE-only Rate-Distortion Trade-off

In the first experiment, the model is trained from scratch for 5000 steps with MSE as the only distortion term, and the rate weight λ is varied over {0.003,0.01,0.03,0.06}. Table 1 below reports training and validation PSNR, bpp, MSE, MS-SSIM, and training time for each λ.

Lambda	TRAIN				VAL				Train Time
	PSNR (dB)	BPP	MS-SSIM	LPIPS	PSNR (dB)	BPP	MS-SSIM	LPIPS	
0.003	28.74	0.529	0.9792	0.1926	28.29	0.535	0.9792	0.1849	2:08
0.01	28.04	0.491	0.9743	0.2181	27.62	0.493	0.9741	0.2099	2:02
0.03	27.08	0.467	0.9674	0.2581	26.61	0.469	0.9669	0.2540	2:03
0.06	26.10	0.468	0.9552	0.3045	25.61	0.469	0.9543	0.2983	2:13

Table 1: Rate–distortion results for different values of λ.

On the validation set, decreasing λ from 0.06 to 0.003 increases PSNR but also increases bpp, illustrating the trade-off when choosing λ; values that are too large eventually worsen all metrics. MS-SSIM similarly improves while LPIPS decreases, indicating better perceptual quality at higher rates. The λ = 0.01 model provides a good compromise, with validation PSNR 27.62 dB at 0.493 bpp, and is used as the baseline for future experiments. As bpp decreases, PSNR and the perceptual metrics rise, confirming that the model performs like a standard rate-distortion system.

4.2 Experiment 2 Adding MS-SSIM to the Loss

In the second experiment, the model from experiment 1 with λ = 0.01 is used as a starting point, and it is fine-tuned for 2000 steps with MS-SSIM added to the loss with weight β varied over $(0, 10^{-4}, 10^{-3}, 5 \cdot 10^{-3})$. This is done because training a pure MS-SSIM model from scratch led to much worse results, with PSNR only reaching 23 dB. Interestingly, MS-SSIM does not seem to improve any metrics, as when β is increased, all metrics get worse, including PSNR and all perceptual metrics. The MSE only fine-tune gives the best results on the validation set, and the largest β hurts performance on all levels. The numbers show that as the parameter increases, all three quality metrics move in the wrong direction and that the MSE-only model preserves structure and texture well. As a result β is kept at 0 for future experiments. Table 2 displays the results for this experiment.

4.3 Experiment 3: Adding LPIPS to the Loss

Experiment 3 uses the baseline established in experiment 1, as adding MS-SSIM to the loss only hampered progress. LPIPS is added to the loss with weight γ ranging from 0 to 0.05, and the model

Beta	TRAIN				VAL				Train Time
	PSNR (dB)	BPP	MS-SSIM	LPIPS	PSNR (dB)	BPP	MS-SSIM	LPIPS	
0	29.04	0.187	0.9832	0.1698	28.65	0.189	0.9828	0.1642	0:50
0.0001	28.85	0.192	0.9819	0.1779	28.40	0.194	0.9813	0.1732	0:49
0.0010	28.04	0.204	0.9753	0.2053	27.73	0.206	0.9757	0.1974	0:48
0.0050	26.60	0.217	0.9394	0.3015	26.35	0.219	0.9467	0.2801	0:48

Table 2: Effect of perceptual loss weight β on rate–distortion performance.

is fine-tuned for 2000 steps with the same training setup. Table 3 shows that small values help perceptual quality, as validation PSNR increases to 29.04 dB with a γ of 0.03 compared to 28.65 dB for the baseline, with a marginal bpp increase. This γ gives the best trade-off, with the highest PSNR, highest MS-SSIM, lowest LPIPS, and only a marginal increase in bpp.

Gamma	TRAIN				VAL				Train Time
	PSNR (dB)	BPP	MS-SSIM	LPIPS	PSNR (dB)	BPP	MS-SSIM	LPIPS	
0	29.04	0.187	0.9832	0.1698	28.65	0.189	0.9828	0.1642	0:50
0.01	29.29	0.199	0.9848	0.1397	28.87	0.202	0.9842	0.1372	0:57
0.03	29.33	0.218	0.9851	0.1263	29.04	0.223	0.9845	0.1218	0:57
0.05	29.32	0.218	0.9852	0.1274	28.88	0.223	0.9844	0.1273	0:57

Table 3: Effect of perceptual loss weight γ on rate–distortion performance.

4.4 Experiment 4: Hyperprior vs Factorized Prior

To study the effect of a more expressive entropy model, the factorized Gaussian prior is replaced with a simple hyperprior, while keeping the same structure, training for 5000 steps with the pure MSE model and then 2000 steps with a β of 0.03. Table 4 compares the two priors. As seen below, the

Prior	TRAIN				VAL				Train Time
	PSNR (dB)	BPP	MS-SSIM	LPIPS	PSNR (dB)	BPP	MS-SSIM	LPIPS	
Factorized	29.33	0.218	0.9851	0.1263	29.04	0.223	0.9845	0.1218	2:59
Hyper	28.44	1.055	0.9800	0.1625	27.97	1.063	0.9793	0.1636	2:57

Table 4: Comparison of Factorized vs. Hyperprior models.

factorized prior's metrics are much better than the hyperprior's on all fronts with almost identical training time. In theory, a well-designed hyperprior should reduce the true rate via modeling spatial variability better, but in this experiment the hyperprior is simpler and accounts for the main latents and hyper-latents, increasing the calculated bpp. In this implementation, the factorized prior performs better with regards to perceptual quality and compression, so it is used for downstream JPEG/WebP comparisons. This winning model also is shown to have 1.95M trainable parameters, very small for image compression, but it only takes 5 ms to encode and decode a 256 x 256 image.

4.5 Experiment 5: JPEG and WebP Comparison Using Kodak

Next, the best learned model is evaluated on an external test set from Kodak24 and compared to JPEG and WebP at three quality levels: 50, 75, and 95. At all qualities, the learned model's bpp is essentially fixed at 0.204 with PSNR 30.57 dB, MS-SSIM 0.9833, and LPIPS 0.1705. For comparison:

- **JPEG Q=75:** 1.558 bpp, 35.20 dB, MS-SSIM 0.9902, LPIPS 0.1336.
- **WebP Q=75:** 0.939 bpp, 34.87 dB, MS-SSIM 0.9829, LPIPS 0.1800.
- **JPEG Q=50:** 0.993 bpp, 32.68 dB, MS-SSIM 0.9813, LPIPS 0.2047.
- **WebP Q=50:** 0.673 bpp, 33.01 dB, MS-SSIM 0.9747, LPIPS 0.2333.
- **JPEG/WebP Q=95:** high quality (PSNR > 41 dB, MS-SSIM > 0.99) at bpp 2.8+

These results show that codecs still win when comparing pure quality given more bits, but when quality is lowered to focus on compression, WebP takes 3 times as many bits to achieve a similar PSNR, although these codecs show a real and practical rate compared to the idealized model bound.

4.6 Experiment 6: Matched-BPP Visual Crops

Lastly, visual crops at matched bpp are presented to understand qualitative differences between codecs. A target bpp close to the learned model's estimate is selected, and JPEG/WebP qualities are chosen so that their file bpp nears this value. Two images are randomly chosen and shown side by side for the original image, the model reconstruction, and JPEG and WebP at the selected quality.

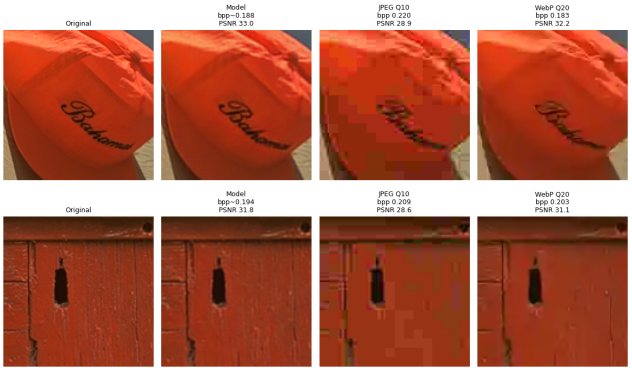

Figure 1: Matched-BPP Visual Crops for a Hat and Wall

The examples show that at a similar nominal bpp, the model outperforms in PSNR, along with the other perceptual metrics. All three compressed versions have some blur, with JPEG noticeably being the worst with more blockiness and roughness, WebP showing a bit of a "washed-out" and too smooth texture, and the learned model showing some loss of fine details. But it is clear visually and through metrics that the learned model performs the best under large compression, with WebP not too far behind. Overall, the experiments show a complete picture of how the proposed learned compressor behaves under different loss weights and entropy models, and how it compares to widely used codecs.

5 Conclusion

This project implemented a compact image compression model using a learned entropy model, perceptual losses, and comparisons to standard codecs. Starting from an MSE-only rate–distortion setup, the experiments showed that a small autoencoder with a factorized Gaussian prior can achieve 28.65 dB validation PSNR with strong perceptual metrics at roughly 0.2 Gaussian proxy bpp. Adding MS-SSIM directly into the loss consistently hurt performance in this setting, as it pulls loss in a different direction than MSE, since it is more tolerant to small shifts in contrast and is more concerned with matching pixel value. Adding a modest LPIPS term on top of MSE slightly increased the estimated rate but improved all three quality metrics and gave qualitatively sharper reconstructions.

The comparison between entropy models and codecs highlights important limitations and opportunities. The factorized Gaussian prior outperformed the simple hyperprior variant in estimated rate and reconstruction quality, as the hyperprior introduced additional bits for its own latents without enough modeling gain. The learned model operated at a much lower estimated bpp on the test set compared to the codecs, but it did not beat them in PSNR or perceptual metrics when higher bitrates were allowed, an understandable outcome with the model's focus on compression. This reflects both the optimistic nature of the model-based bpp estimate and how effective hand-engineered codecs are. Overall, the results suggest that small learned compressors can be competitive in the very low–rate regime and that LPIPS-style perceptual terms are more promising than MS-SSIM in this setup. Future work could include a more advanced hyperprior or joint auto regressive prior with a true entropy coder and exploring adaptive-rate models that can better match codecs across a wide range of bitrates.

References

[1] Ballé, J., Laparra, V., & Simoncelli, E. P. (2017). *End-to-end optimized image compression*. arXiv. https://arxiv.org/abs/1611.01704

[2] Ballé, J., Minnen, D., Singh, S., Hwang, S. J., & Johnston, N. (2018). *Variational image compression with a scale hyperprior*. arXiv. https://arxiv.org/abs/1802.01436