LOSSLESS IMAGE CODING USING CONTEXT-DRIVEN NEURAL DISTRIBUTION ESTIMATION

Victor F. Figueiredo[1]*, Lucas S. Lopes*[1]*, Ricardo L. de Queiroz*[1] *and Philip A. Chou*

[1]Universidade de Brasília, Brasília, Brazil

ABSTRACT

We present an innovative method for lossless coding using prediction and context-adaptive Laplacian modeling to encode the residuals. As far as we know, this is the new state-of-the-art in lossless image compression. Given a causal context, a lightweight neural network predicts the signal mean, while another estimates the scale of the prediction residual, assuming a zero-mean Laplacian density for the residual. The density is integrated to estimate a probability of the quantized residual, which drives an adaptive arithmetic coder to encode the quantized residual. The quantization of the residual is equivalent to using a shifted Laplacian to drive an arithmetic coder to code the original quantized signal. The networks are jointly trained to minimize the resulting bit rate. We instantiate different architectures to evaluate the method and results show that the best, yet simple, architecture variant (a multilayer perceptron) outperforms the competition, reducing the average bit rate by 30.58% relative to HEVC Intra Lossless and outperforming the state-of-the-art, on average, while remaining shallow enough for real-time operation. These results indicate that jointly predicting the mean of the signal and scale of the prediction residuals enables efficient lossless coding, outperforming the current state-of-the-art lossless image codec.

***Index Terms*—** Entropy bottleneck, lossless image compression, context modeling, neural networks.

1. INTRODUCTION

Lossless image compression remains an important problem. Traditional approaches leverage statistical redundancy by predicting pixel values based on causal neighborhood contexts, significantly improving compression performance during entropy coding stages [1, 2, 3, 4]. More recently, neural-network-based methods have further advanced compression efficiency through sophisticated modeling of pixel prediction errors and of context dependencies [5, 6, 7, 8, 9].

Standard lossless image compression codecs like the H.265/HEVC video codec operating in intra-frame lossless mode [10] (HEVC Intra Lossless) and classical context-based adaptive techniques such as CALIC [1, 2] were significantly outperformed by neural-network-based methods, such as convolutional neural networks (CNNs) and deep autoregressive models, that are effective in capturing complex spatial dependencies [8].

The work presented in Context-based Bit-plane Codec (CBPNNv) [8] is considered the current state-of-the-art for lossless image compression. In it, it is proposed a lossless pipeline that couples a classical spatial predictor with a CNN-based residual refiner and a CALIC-style context model feeding a context-based bit-plane (CBP) arithmetic coder. The causal input to the CNN is a wide stencil and the network outputs an 8-bit residual class only used to refine the classical prediction, preserving exact invertibility. After prediction, the integer residual is modeled by a CALIC-inspired context that combines a local binary pattern with a quantized energy term to form an index. Residuals are mean-shifted per context and remapped using the current predictor output to sharpen entropy modeling. The refined error is then encoded by a CBP codec that first predicts the binary length, signals an under-prediction flag when needed, and arithmetic-codes each bit-plane using adaptive-context trees with periodic count halving.

In popular neural-based schemes for lossy image compression [5], [6, 11] there is often a layer to handle both quantization and probability estimation during the training phase. This is sometimes referred to as the Entropy Bottleneck layer [12, 13]. A principal innovation in the Entropy Bottleneck layer is the so-called hyperprior [6, 11], which sends side information to adapt the variances of the neural coefficients to different spatial contexts. The hyperprior is a modern take on decades-old solutions to spatial adaptation of the variances for efficient coding such as [14].

In this work, for lossless image compression, we propose two neural networks, one for prediction and one analogous to a hyperprior network, but not requiring side information. Specifically, given a causal context, a prediction network outputs real-valued predictions of the integerized pixel to be encoded. The prediction error, or residual, is then quantized and entropy coded. Dithered quantization of the residual is used, so that the quantization bin of the residual corresponds to that of the original integerized pixel. The probability of the quan-

Work partially supported by CNPq under grant 302957/2022-7 and by Coordenação de Aperfeiçoamento de Pessoal Superior - Brasil (CAPES) - Finance Code 001

tized residual is estimated by integrating a zero-mean Laplacian density over the quantization bin [15, 16]. The scale of the zero-mean Laplacian density is provided by a second network, given the same causal context, making it adaptive. The probability derived from the density drives an adaptive arithmetic coder. The networks are jointly trained to minimize the bit rate of the arithmetic coder. We show that this coding method is equivalent to directly entropy coding the integerized pixel with a shifted Laplacian distribution whose shift and scale are given by the prediction and scale networks. Our approach easily extends to other distributions with more parameters.

Our lossless image compression approach significantly reduces bit-stream sizes compared to traditional lossless image compression methods [10, 3] and to more recently proposed deep-learning-based methods [8]. Indeed, experimental validation is carried on the Ultra Video Group (UVG) dataset [17] and results show that our approach consistently outperforms reference methods such as HEVC Intra Lossless, CALIC, and the state-of-the-art [8].

2. PROPOSED CODER

2.1. Entropy Model and Coder

Let x_{ij} denote the value of the pixel in the i-th row and j-th column of a b-bit monochrome image. We assume that x_{ij} lies in the range $[0, 2^b)$, but is truncated to an integer $k_{ij} = \lfloor x_{ij} \rfloor$ in the range $\{0, \ldots, 2^{b-1}\}$. The problem is to encode and transmit k_{ij} losslessly for all ij.

We let $\mathcal{C}_{ij}$ be the causal context of x_{ij} comprising its N closest causal pixels. We determine the N closest causal pixels geometrically assuming square pixels. Figure 1 depicts the selected causal context for $N = 32$. We assume the integerized pixels in $\mathcal{C}_{ij}$ have been previously encoded and are thus available to the decoder prior to decoding k_{ij}. If some pixels in $\mathcal{C}_{ij}$ lie outside the image boundaries, we assume they have a fixed constant value.

We let $\mu_{ij} = \mu(\mathcal{C}_{ij})$ be a real-valued prediction of x_{ij} based on the context $\mathcal{C}_{ij}$. We will implement the function $\mu : \mathcal{C}_{ij} \mapsto \mu_{ij}$ with a neural network, as described in the next subsection.

We let r_{ij} be the prediction error, or residual, where

$$r_{ij} = x_{ij} - \mu_{ij}. \tag{1}$$

We model the distribution of r_{ij} as a zero-mean Laplacian with density $f(r; 0, \beta_{ij})$, where

$$f(r; 0, \beta) = \frac{1}{2\beta} \exp\left(-\frac{|r|}{\beta}\right) \tag{2}$$

and $\beta_{ij} = \beta(\mathcal{C}_{ij})$ is the *scale* of the zero-mean Laplacian distribution.[1] We will implement the function $\beta : \mathcal{C}_{ij} \mapsto \beta_{ij}$ with a neural network, as described in the next subsection.

[1] β_{ij} is also known as the *mean absolute deviation* or the *standard deviation* (divided by $\sqrt{2}$) of the Laplacian distribution.

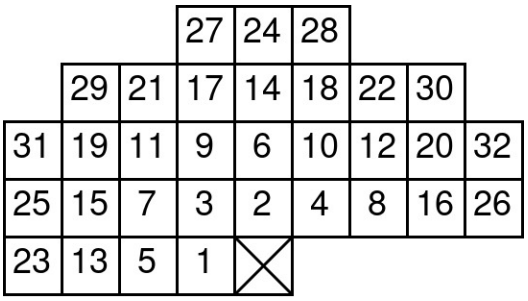

Fig. 1. Example of the causal context $\mathcal{C}_{ij}$ for the current pixel x_{ij} (marked with $\times$). Candidate causal pixels (i.e., previously decoded pixels in raster-scan order) are ranked by geometric proximity under the square-pixel assumption; the labels indicate the order of increasing distance, from which the N closest causal neighbors are selected. In this example $N = 32$.

Equivalently, we model the distribution of x_{ij} as a shifted Laplacian with density $f(x; \mu_{ij}, \beta_{ij})$, where

$$f(x; \mu, \beta) = \frac{1}{2\beta} \exp\left(-\frac{|x - \mu|}{\beta}\right) \tag{3}$$

and μ_{ij} is the *mean* of the Laplacian distribution.[2] Thus μ_{ij} may be interpreted either as the prediction of x_{ij} or as the mean of the Laplacian distribution that models the density of x_{ij}. Likewise β_{ij} may be interpreted either as the scale of the zero-mean Laplacian distribution of r_{ij} or as the scale of the mean-μ_{ij} Laplacian distribution of x_{ij}.

Using an arithmetic coder, we transmit $k_{ij} = \lfloor x_{ij} \rfloor$ using $-\log_2 p(k_{ij}; \mu_{ij}, \beta_{ij})$ bits, where $p(k; \mu, \beta)$ is the integral of the density (3) over bin k,

$$p(k; \mu, \beta) = \int_k^{k+1} \frac{1}{2\beta} \exp\Big(-\tfrac{|x-\mu|}{\beta}\Big) dx \tag{4}$$

$$= F(k+1; \mu, \beta) - F(k; \mu, \beta), \tag{5}$$

where $F(x; \mu, \beta)$ is the cumulative distribution function

$$F(x; \mu, \beta) = \int_{-\infty}^{x} f(t; \mu, \beta) dt \tag{6}$$

$$= \begin{cases} \frac{1}{2}\exp(-(x-\mu)/\beta) & \text{if } x \geq \mu \\ 1 - \frac{1}{2}\exp((\mu - x)/\beta) & \text{if } x \leq \mu \end{cases}. \tag{7}$$

Thus we have

$$-\log_2 p(k; \mu, \beta) \tag{8}$$

$$= -\log_2(F(k+1; \mu, \beta) - F(k; \mu, \beta)) \tag{9}$$

$$\approx \frac{1}{\ln 2}\left(\tfrac{|k+1/2-\mu|}{\beta} + \ln\big(2\beta\big)\right). \tag{10}$$

[2] μ_{ij} is also known as the *shift* or *center* of the distribution.

Our objective is to minimize the total number of bits

$$R = -\sum_{ij} \log_2 p(k_{ij}; \mu(\mathcal{C}_{ij}), \beta(\mathcal{C}_{ij})). \tag{11}$$

We therefore jointly train the prediction network $\mu(\mathcal{C})$ and the scale network $\beta(\mathcal{C})$ to minimize (11) over a representative training set. For the purposes of training, one may use the approximation (10) if $\beta \gg 1$, but the exact expression (9) is preferred. Note that (9) is still differentiable almost everywhere (as is ReLU, for example). Thus we may train the networks either simultaneously or alternately using back-propagation and gradient descent through the loss function (11).[3]

Thus the predictor network $\mu(\mathcal{C})$ and the scale network $\beta(\mathcal{C})$ can be regarded as networks that determine the parameters of the density used to code x_{ij}. From this point of view, the encoder and decoder communicate k_{ij} directly.

In an alternative and equivalent point of view, the encoder and decoder may independently determine the prediction $\mu_{ij} = \mu(\mathcal{C})$, and then communicate the quantized prediction residual

$$\hat{r}_{ij} = k_{ij} - \mu_{ij}. \tag{12}$$

Specifically, the encoder may 1) use the predictor network to predict x_{ij} as $\mu_{ij} = \mu(\mathcal{C}_{ij})$, 2) subtract μ_{ij} from $k_{ij} = \lfloor x_{ij} \rfloor = \lfloor \mu_{ij} + r_{ij} \rfloor$ to obtain the dithered[4] quantized residual $\hat{r}_{ij} = \lfloor \mu_{ij} + r_{ij} \rfloor - \mu_{ij}$, and 3) transmit $\hat{r}_{ij}$ using $-\log_2 p(\hat{r}_{ij}; 0, \beta_{ij})$ bits, where

$$p(\hat{r}; 0, \beta) = F(\hat{r} + 1; 0, \beta) - F(\hat{r}; 0, \beta) \tag{13}$$

and $\beta_{ij} = \beta(\mathcal{C}_{ij})$. Then, the decoder may 1) again use the predictor network to predict x_{ij} as $\mu_{ij} = \mu(\mathcal{C}_{ij})$, 2) decode $\hat{r}_{ij}$ using the integral of the zero-mean Laplacian density (13), where the scale parameter is determined by the scale network $\beta_{ij} = \beta(\mathcal{C}_{ij})$, and 3) recover $k_{ij} = \mu_{ij} + \hat{r}_{ij}$.

These two approaches are equivalent because they use equivalent codes produced by identical probabilities:

$$p(\hat{r}; 0, \beta) = \int_{\hat{r}}^{\hat{r}+1} \frac{1}{2\beta} \exp\Big(-\tfrac{|r|}{\beta}\Big) dr \tag{14}$$

$$= \int_{\hat{r}+\mu}^{\hat{r}+\mu+1} \frac{1}{2\beta} \exp\Big(-\tfrac{|x-\mu|}{\beta}\Big) dx \tag{15}$$

$$= \int_{k}^{k+1} \frac{1}{2\beta} \exp\Big(-\tfrac{|x-\mu|}{\beta}\Big) dx \tag{16}$$

$$= p(k; \mu, \beta). \tag{17}$$

Dithering the quantization of r_{ij}, so that $\hat{r} = k - \mu$ (whereas $r = x - \mu$), is what makes these exact. Note that $\hat{r}$ is not generally an integer. However, its quantization bin aligns with the quantization bin of the integer $k = \lfloor x \rfloor$, and hence the decoder can losslessly recover k from the quantization bin. This trick would not be necessary for lossy compression.

Figure 2 shows the estimated zero-mean Laplacian distribution for pixel x_{ij}. Figure 3 illustrates the proposed equivalent coding systems.

Our approach may be extended easily to distributions with more parameters, for example the generalized Gaussian distribution or distributions with a learned parameterization.

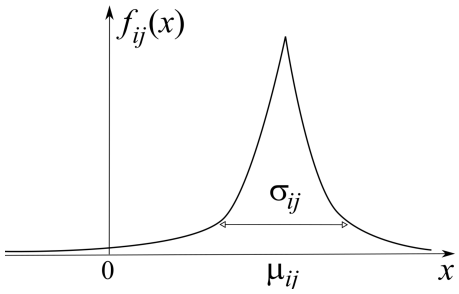

Fig. 2. Illustration of the estimated distribution using causal context $\mathcal{C}_{ij}$ for pixel x_{ij}. It is important to notice that for each pixel a unique distribution is estimated based on its context, what they all have in common is that they are shifted Laplacians. $f_{ij}(x_{ij}; \mu_{ij}, \beta_{ij})$ is the estimated distribution, where μ_{ij} is the mean, σ_{ij} is the standard deviation and $\sigma_{ij} = \sqrt{\beta_{ij}}$.

2.2. Network Architectures

We implemented both the prediction and the scale networks using three different neural architectures: a lightweight Convolutional Neural Network (CNN), a Multi-Layer Perceptron (MLP), and a lightweight Transformer Network. For all network architectures, the prediction and scale networks were trained alternately to convergence.

The CNN baseline maps a single-channel context patch of spatial size $(N+2) \times (N+2)$ to a scalar output. It consists of two 3×3 convolutional layers with padding 1, increasing the number of channels from 1 to 16 and then to 32, each followed by a ReLU nonlinearity. The resulting feature map is flattened and passed through a fully connected layer of size $32(N+2)^2 \to 64$ with ReLU, followed by a final linear layer of size $64 \to 1$. Including biases, this architecture contains $160 + 4640 + \big(32(N+2)^2 \cdot 64 + 64\big) + 65 = 2048(N+2)^2 + 4929$ trainable parameters, which yields 299 841 parameters for $N = 10$.

The MLP architecture operates on an N-dimensional causal context vector and consists of three affine layers with widths $N \to 64N \to 32N \to 1$, with ReLU activations after the first two layers and a final sigmoid nonlinearity. Including biases, the total number of trainable parameters is $(N \cdot 64N + 64N) + (64N \cdot 32N + 32N) + (32N \cdot 1 + 1) =$

[3] In order to use b-bit pixels with neural networks, we normalize the range of the inputs and outputs from $\{0, \ldots, 2^b - 1\}$ to $[0, 1]$. That is, $\mu(\mathcal{C}) = 2^{b-1}\bar{\mu}(\mathcal{C}/2^{b-1})$ and $\beta(\mathcal{C}) = 2^{b-1}\bar{\beta}(\mathcal{C}/2^{b-1})$ and we train $\bar{\mu}$ and $\bar{\beta}$.

[4] A *dithered* quantizer is one that adds a value before quantization and subtracts it afterwards, effectively shifting the quantization bin.

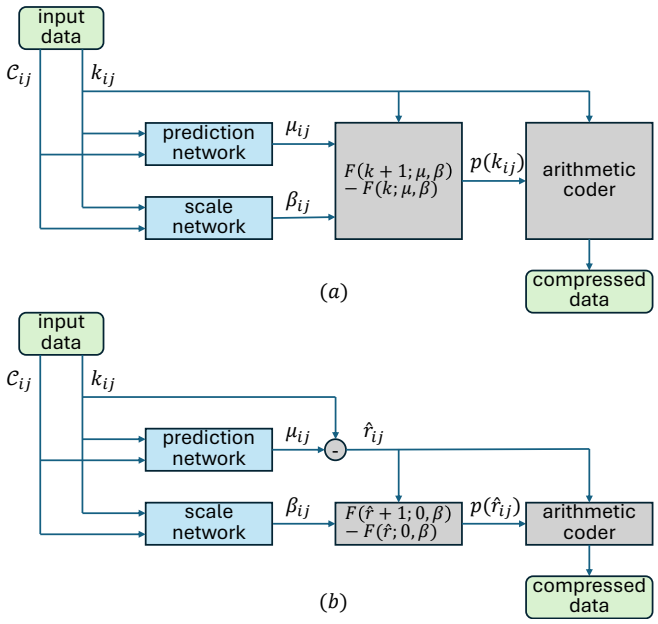

Fig. 3. Diagram illustrating proposed equivalent coding systems, where (a) prediction and scale networks output parameters of Laplacian(μ_{ij}, β_{ij}) distribution, which drives the arithmetic coder to code the pixel value k_{ij} directly, and (b) prediction network outputs prediction μ_{ij}, and scale network produces parameter of Laplacian$(0, \beta_{ij})$ distribution, which drives the arithmetic coder to code the prediction residual $\hat{r}_{ij}$.

$2112N^2 + 128N + 1$, which corresponds to 212 481 parameters for $N = 10$.

The Transformer-based network maps a sequence of N scalar pixel values, quantized to a 256-level grayscale alphabet, to a scalar output in $(0, 1)$ that parameterizes the modeled distribution. Each input token is embedded into a 20-dimensional space via the sum of learned token embeddings (of size 256×20) and positional embeddings (of size $N \times 20$). The embedded sequence is processed by a single transformer block comprising a 2-head self-attention layer with model dimension 20 and a position-wise feed-forward module with dimensions $20 \rightarrow 40 \rightarrow 20$, with both sub-layers equipped with residual connections, layer normalization, and dropout with probability 0.13. The output sequence is then averaged over the temporal dimension and fed to a classifier head implemented as a three-layer perceptron with dimensions $20 \rightarrow 20 \rightarrow 20 \rightarrow 1$, LeakyReLU nonlinearities, dropout, and a final sigmoid activation. For this configuration, the total number of trainable parameters is $20N + 9521$, i.e., 10 521 parameters for $N = 50$.

The CNN and MLP architectures thus have comparable capacity for the same N, while the Transformer variant is deliberately kept lightweight. Additionally, all networks use a causal context of up to $N = 50$ pixels. Each architecture is instantiated twice, one instance for pixel prediction and another for scale estimation. Both networks are separate, but share their input, the causal context.

3. EXPERIMENTAL RESULTS

In our tests, we used the Ultra Video Group (UVG) dataset [17], which is made of 16 versatile 4K (3840 x 2160) 8-bit video sequences. We also use the Gray 8-bit Test Images dataset [18], which is comprised of 15 gray high-resolution photographic images. For our training, we used 9 sequences from the UVG dataset, while the other 7 are used for testing. The training sequences were CityAlley, FlowerFocus, FlowerKids, FlowerPan, Lips, RaceNight, RiverBank, SunBath and Twilight. Such a dataset was meant to coincide with that used in [8].

All experiments were run on a single NVIDIA GPU (RTX 3080) and the memory footprint of our models is sufficiently small to allow training on most modern GPUs. They are intentionally kept shallow and light to approximate real-time performance for high-resolution images.

Table 1. Bit-rate results of our proposed method using the MLP architecture compared to some state-of-the-art codecs evaluated on the Gray 8-bit Test Images dataset.

	State-of-the-art codecs			Proposed Method
Image	**JPEG-LS**	**JPEG2000**	**FLIF**	**MLP**
artificial	0.798	1.191	**0.620**	0.866
big_building	3.592	3.649	**3.447**	3.451
big_tree	3.732	3.799	3.621	**3.402**
bridge	4.148	4.189	4.086	**3.620**
cathedral	3.570	3.704	**3.453**	4.085
deer	4.659	4.577	4.505	**4.503**
fireworks	1.465	1.649	1.359	**1.358**
flower_foveon	2.038	2.193	1.892	**1.886**
hdr	2.175	2.339	2.068	**2.064**
leaves_iso_1600	4.486	4.675	3.687	**3.320**
leaves_iso_200	**3.820**	4.076	4.412	4.316
nightshot_iso_100	2.129	2.295	1.964	**1.961**
nightshot_iso_1600	3.971	4.032	3.852	**3.848**
spider_web	1.766	1.905	1.573	**1.572**
zone_plate	7.429	5.749	4.385	**4.382**
Average bpp	*3.319*	*3.335*	*2.995*	**2.976**

On the Gray 8-bit Test Images dataset, we evaluate the performance of the proposed coder against (i) JPEG-LS [19, 20], (ii) JPEG 2000 [21], and (iii) FLIF [3]. For this experiment, we report results only for the MLP-based variant, as it achieved the best performance among our architectures on the UVG dataset. We do not compare against the CBPNNv for this dataset, since the results for the UVG dataset were the only available in a public dataset. The results are summarized in Table 1, enabling a direct comparison with these state-of-the-art codecs.

On the UVG dataset, even though we are using video sequences to run our tests, there is no inter-frame prediction of any kind. All the tests were run on the luminance channel. The experimental evaluation compares our coder performance to: (i) the HEVC Intra Lossless coder [10] within the

Table 2. Bit-rate results of our proposed method using the three architectures compared to some state-of-the-art codecs evaluated on the UVG dataset. Bit-rate savings (in %) against HEVC Intra Lossless also indicated.

	State-of-the-art codecs				Proposed method		
Video Sequence	**HEVC Intra Lossless**	**FLIF**	**CALIC**	**CBPNNv**	**CNN**	**MLP**	**Transformer**
Beauty	3.76	3.45 (8.14%)	3.42 (8.91%)	3.40 (9.61%)	2.22 (41.06%)	**2.08 (44.56%)**	3.39 (9.81%)
Bosphorus	3.12	2.48 (20.65%)	2.43 (22.15%)	2.38 (23.73%)	2.90 (7.08%)	**2.37 (23.91%)**	2.92 (6.45%)
HoneyBee	3.57	3.01 (15.62%)	2.97 (16.69%)	2.94 (17.44%)	2.70 (24.25%)	**2.59 (27.48%)**	2.94 (17.63%)
Jockey	3.18	2.76 (13.10%)	2.75 (13.39%)	2.71 (14.80%)	2.51 (20.87%)	**2.38 (25.15%)**	2.73 (13.94%)
ReadySteadyGo	3.47	2.75 (20.70%)	2.68 (22.72%)	2.62 (24.62%)	2.85 (17.77%)	**2.33 (32.82%)**	2.63 (24.24%)
ShakeNDry	4.16	3.23 (22.31%)	3.15 (24.28%)	3.15 (24.25%)	2.67 (35.80%)	**2.65 (36.22%)**	3.07 (26.19%)
YachtRide	3.41	2.65 (22.19%)	2.56 (24.89%)	**2.51 (26.53%)**	3.14 (7.99%)	2.59 (23.94%)	3.18 (6.88%)
Average bpp	*3.52*	*2.90 (17.53%)*	*2.85 (19.00%)*	*2.81 (20.14%)*	*2.71 (22.12%)*	**2.43 (30.58%)**	*2.98 (15.02%)*

x265 implementation, operating with the "veryslow" lossless-mode preset, and only using intra prediction; (ii) FLIF [3]; (iii) CALIC [2]; and (iv) CBPNNv under the SLOW profile [8]. Table 2 shows the results, where we can compare the proposed method against the above-mentioned state-of-the-art methods. Along with the rate, we also quote the percentage of bit-rate savings compared to the HEVC Intra Lossless method.

Note that all the three proposed architectures outperformed the HEVC Intra Lossless. The MLP architecture, however, is the one that yields the best results. The proposed method using the MLP architecture outperforms the HEVC Intra Lossless on average by 30.58%. It also outperforms the CBPNNv method on average by 13.71%. Hence, we believe that our method using MLP is the new state-of-the-art in lossless image compression.

4. CONCLUSIONS

This work presents a novel approach to lossless image coding by introducing a neural-network-based context modeling framework that uses both the mean and scale of the prediction error to improve the arithmetic coding stage of the final bitstream. Its design avoids side information while retaining the benefits of learned, spatially varying scale.

Our approach is equivalent to using neural networks to select the parameters of a probability distribution over the input symbol for each context. As such, it easily extends to probability distributions with multiple parameters.

We also present a comparison between different network architectures to determine the best suited architecture for a shallow and light neural network to facilitate real-time or near-real-time performance. Experimental results indicate that the proposed method outperforms the state-of-the-art [8] on the same dataset used in that work.

We plan to further investigate the use of residual distribution estimation to improve the arithmetic coding stage in lossy image compression. We may also extend our approach to other media such as videos and point clouds, as well as to distributions with multiple parameters.

5. REFERENCES

[1] X. Wu and N. Memon, "CALIC-a context based adaptive lossless image codec," in *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*, 1996, vol. 4, pp. 1890–1893 vol. 4.

[2] X. Wu and N. Memon, "Context-based, adaptive, lossless image coding," *IEEE Transactions on Communications*, vol. 45, no. 4, pp. 437–444, 1997.

[3] J. Sneyers and P. Wuille, "FLIF: Free lossless image format based on MANIAC compression," in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 66–70.

[4] I. Matsuda, T. Ishikawa, Y. Kameda, and S. Itoh, "A lossless image coding method based on probability model optimization," in *2018 26th European Signal Processing Conference (EUSIPCO)*, 2018, pp. 151–155.

[5] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

[6] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a hyperprior," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[7] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. 2018, vol. 31, Curran Associates, Inc.

[8] I. Schiopu and A. Munteanu, "Deep-learning-based lossless image coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 7, pp. 1829–1842, 2020.

[9] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learned image compression with discretized gaussian mixture likelihoods and attention modules," in *2020 IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 7936–7945.

[10] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[11] J. Ballé, P. A. Chou, D. Minnen, S. Singh, N. Johnston, E. Agustsson, S. J. Hwang, and G. Toderici, "Nonlinear transform coding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 339–353, 2021.

[12] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," 2018.

[13] J. Bégaint, F. Racapé, S. Feltman, and A. Pushparaja, "Compressai: a pytorch library and evaluation platform for end-to-end compression research," *arXiv preprint arXiv:2011.03029*, 2020.

[14] S. LoPresto, K. Ramchandran, and M. Orchard, "Image coding based on mixture modeling of wavelet coefficients and a fast estimation-quantization framework," in *Proceedings DCC '97. Data Compression Conference*, 1997, pp. 221–230.

[15] J. B. O'Neal, "Predictive quantizing systems (differential pulse code modulation) for the transmission of television signals," *The Bell System Technical Journal*, vol. 45, no. 5, pp. 689–721, 1966.

[16] G. Langdon and A. Zandi, "Characterizing prediction error distributions for lossless image compression," in *Conference Record of The Thirtieth Asilomar Conference on Signals, Systems and Computers*, 1996, vol. 1, pp. 573–576 vol.1.

[17] A. Mercat, M. Viitanen, and J. Vanne, "UVG dataset: 50/120fps 4K sequences for video codec analysis and development," in *Proceedings of the 11th ACM Multimedia Systems Conference*, New York, NY, USA, 2020, MMSys '20, p. 297–302, Association for Computing Machinery.

[18] Rawzor, "Gray 8-bit Test Images (The New Test Images)," [Online]. Available: `https://imagecompression.info/test_images/`, 2015, Accessed: 2026-01-29.

[19] M. Weinberger, G. Seroussi, and G. Sapiro, "The loco-i lossless image compression algorithm: principles and standardization into jpeg-ls," *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1309–1324, 2000.

[20] M. Weinberger, G. Seroussi, and G. Sapiro, "From logo-i to the jpeg-ls standard," in *Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)*, 1999, vol. 4, pp. 68–72 vol.4.

[21] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The jpeg 2000 still image compression standard," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 36–58, 2001.