ENTROFORMER: A TRANSFORMER-BASED ENTROPY MODEL FOR LEARNED IMAGE COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

One critical component in lossy deep image compression is the entropy model, which predicts the probability distribution of the quantized latent representation in the encoding and decoding modules. Previous works build entropy models upon convolutional neural networks which are inefficient in capturing global dependencies. In this work, we propose a novel transformer-based entropy model, termed *Entroformer*, to capture long-range dependencies in probability distribution estimation effectively and efficiently. Different from vision transformers in image classification, the Entroformer is highly optimized for image compression, including a top-k self-attention and a diamond relative position encoding. Meanwhile, we further expand this architecture with a parallel bidirectional context model to speed up the decoding process. The experiments show that the Entroformer achieves state-of-the-art performance on image compression while being time-efficient.

1 INTRODUCTION

Image compression is a fundamental research field in computer vision. With the development of deep learning, learned methods have led to several breakthroughs in this task. Currently, the state-of-the-art (SOTA) deep image compression models are built on the auto-encoder framework (Hinton & Salakhutdinov, 2006) with an entropy-constrained bottleneck (Theis et al., 2017; Ballé et al., 2017; Ballé et al., 2018; Mentzer et al., 2018; Minnen et al., 2018a; Lee et al., 2019). In this framework, an encoder converts pixels into quantized latents suitable for compression, while a decoder is jointly optimized to turn latents back into pixels. Then, the latents are losslessly compressed to a bitstream by using the entropy coding method. The entropy model estimates the conditional probability distribution of latents, which is therefore critical to the compression rate. In this paper, we focus on boosting image compression performance through a novel entropy model.

One of the main challenges of the entropy model is *how to learn representation for different dependencies*. Some works use neighbouring symbols (*e.g.*context model) (Minnen et al., 2018a; Lee et al., 2019; Mentzer et al., 2018) or additional local side information (*e.g.*hyperprior module) (Ballé et al., 2018; Hu et al., 2020; Minnen et al., 2018b) as spatial dependencies, while others use non-local mechanism (Li et al., 2020; Qian et al., 2021; Cheng et al., 2020) to capture long-range dependencies. However, when capturing dependencies for probability distribution estimation, constraints still remain in information distillation and representation ability. For example, as shown in Figure 1, the wall has identical texture, which is no reason to be disturbed by the sky; and the five hats are likely to be related in different subspaces of color and shape.

Another main challenge is the *inefficient decoding process*. Though improving the predictive ability, previous context model methods suffer from slow decoding speed. It decodes symbols in a raster-scan order with $O(n)$ serial process that can not be accelerated by modern GPUs. A two-pass parallel context model is introduced for acceleration (He et al., 2021). However, this parallel context model use a weak context information, which degrades the final compression performance.

In this paper, we propose a novel transformer-based entropy model termed as *Entroformer* to address the above two challenges in one shot. In Entroformer, spatial and content-based dependencies are taken into account in both hyperprior and context model. The self-attention module automatically learns the global dependencies in a transformer-like structure. To further optimize Entroformer for image compression, the following designs are innovated: (1) Entroformer uses a multi-head

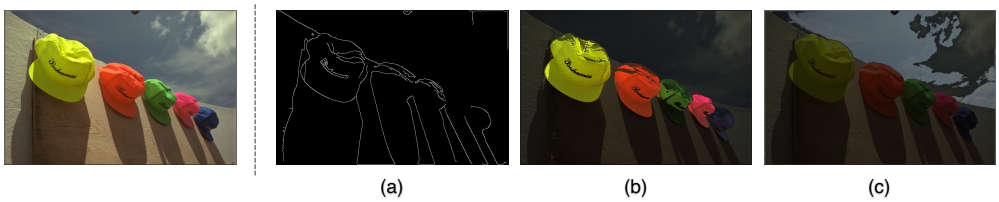

Figure 1: Spatial and content dependencies exist in (a) texture, (b) semantic and (c) color space.

attention with a top-k selection to distill information in representation subspaces: The multi-head attention provides a learning-based partition to discover dependencies in different views, and the top-k scheme allows the most similar k latents to the self-attention module, filtering noisy signals. (2) A novel position encoding unit is designed to provide better spatial representation for the self-attention module. This position encoding is based on a relative position encoding with a diamond-shape boundary, which takes full advantage of prior information for image compression. (3) The two-pass decoding framework (He et al., 2021) is utilized to speed up the decoding of Entroformer. A bidirectional context is introduced instead of the local checkeboard context, which helps counteract the performance degradation. In summary, our contributions include:

- In this paper, Entroformer is proposed to improve learned image compression. To our knowledge, it is the first successful attempt to introduce Transformer based method to image compression task. Experiments show that our method outperforms the most advanced CNNs methods by 5.2% and standard codec BPG (Bellard., 2014) by 20.5% at low bit rates.
- To further optimize Entroformer for image compression, the multi-head scheme with top-k selection is proposed to capture precise dependencies for probability distribution estimation, and the diamond relative position encoding (diamond RPE) is proposed for better positional encoding. Experiments show that these methods support image compression task to keep training stable and obtain superior results.
- With the help of bidirectional context and two-pass decoding framework, our parallel Entroformer is more time-efficient than the serialized one on modern GPU devices without performance degradation.

2 Compression with Hyperprior and Context

In this work, we focus on the challenge of modelling the exact probability density of quantized latents for efficient arithmetic coding. There are types of entropy models such as hyperprior (Minnen et al., 2018b; Ballé et al., 2018), context model (Li et al., 2018; Mentzer et al., 2018), and the combined method (Lee et al., 2019; Minnen et al., 2018a; Minnen & Singh, 2020). Hyperprior methods typically make use of side information of the quantized latent representation with additional bits. Context models learn auto-regressive priors incorporates prediction from the causal context of the latents, which is a bit-free. Our model combines a context model with the hyperprior based on previous works (Minnen et al., 2018b; Ballé et al., 2018).

Typical learned lossy image compression methods are based on autoencoders with constrained entropy. The encoder first maps pixel data x into a latent representation y, which is discretized to form $\hat{y}$. The decoder then forms the reconstructed image $\hat{x}$ from the quantized latent representation $\hat{y}$. The discretized value $\hat{y}$ can be losslessly compressed into a bitstream by a prior probability model $p_{\hat{y}}$ (the entropy model) shared between the encoder and decoder. Hyperprior use hyper-encoder and hyper-decoder to learn a hyper-latent representation z, which is discretized to form $\hat{z}$. The training goal for learned image compression is to optimize the trade-off between the estimated coding length of the bitstream and the quality of the reconstruction, which is a rate-distortion optimization problem:

$$\mathcal{L} = R + \lambda D = \mathbb{E}_{x\sim p_x}[-\log_2 p_{\hat{y}}(\hat{y})] + \mathbb{E}_{x\sim p_x}[-\log_2 p_{\hat{z}}(\hat{z})] + \lambda \mathbb{E}_{x\sim p_x}[d(x, \hat{x})], \tag{1}$$

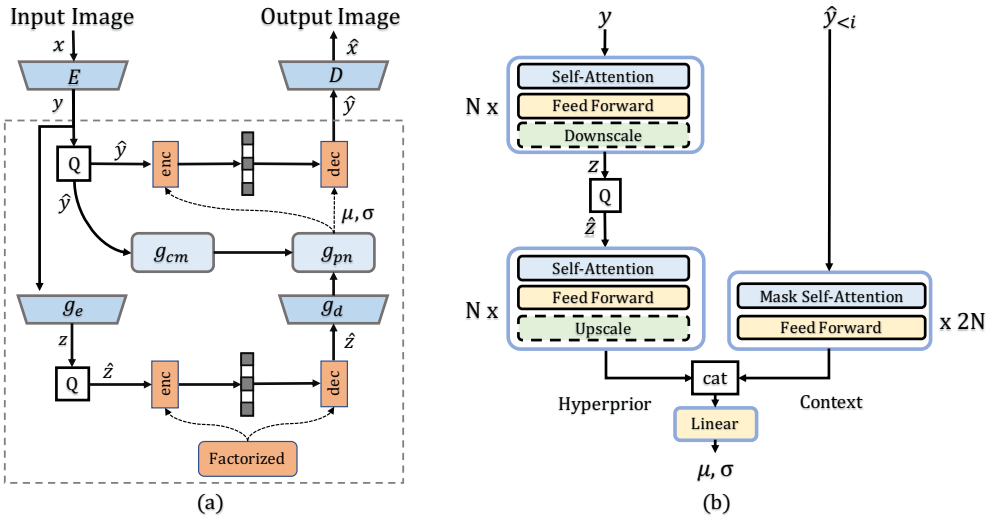

Figure 2: In (a), we show the architecture of image compression model with hyerprprior and context model. The main autoencoder (E, D) learns an latent representation of image, while a hyper-autoencoder (g_e,g_d) learns a hyperprior representation. Context model (g_{cm}) learns correlation of latents from their decoded context. Real-valued latent representation (y, z) is quantized (Q) to create latents ($\hat{y}$) and hyper-latents ($\hat{z}$), which are compressed into a bitstream using an arithmetic encoder and decompressed by an arithmetic decoder. The latents use a Gaussian entropy model conditioned on hyperprior and context. The region in dashed line corresponds to our transformer-based entropy model, which is shown in (b). A transformer encoder learns the hyerprprior, while a transformer decoder learns the context using masked self-attention. Linear component is used to joint these features to generate the Gaussian parameters (*i.e.* μ, σ).

where λ is the coefficient which controls the rate-distortion trade-off, p_x is the unknown distribution of natural images. $p_{\hat{z}}$ use a fully factorized density model. The first term represents the estimated compression rate of the latent representation. The second term $d(x, \hat{x})$ represents the distortion value under given metric, such as mean squared error (MSE).

Following the work of Ballé et al. (2016), the quantization operation is approximated by a additive uniform noise during training. For simplicity, we always denote the discretized quantity and the approximated quantity with a hat. We model each latent, $\hat{y}_i$, as a Gaussian with mean μ_i and deviation σ_i convolved with a unit uniform distribution:

$$p_{\hat{y}}(\hat{y}|\hat{z}, \theta) = \prod_{i=1} (\mathcal{N}(\mu_i, \sigma_i^2) * \mathcal{U}(-0.5, 0.5)))\,(\hat{y}_i) \tag{2}$$

where μ and σ are predicted by the entropy model, θ is the parameters of the entropy model. The entropy model consists of hyper-autoencoder (g_e,g_d), context model (g_{cm}), and parameter networks (g_{pn}):

$$\mu_i, \sigma_i = g_{pn}(g_d(\hat{z}), g_{cm}(\hat{y}_{<i})) \tag{3}$$

where $\hat{y}_{<i}$ represents the causal context of $\hat{y}_i$. In previous unidirectional context model, prediction for position i depends only on the decoded latents above and to the left. Our bidirectional context model allows each prediction to access both left and right context $\hat{y}_{<i}$ and $\hat{y}_{>i}$.

3 Transformer-based Entropy Model

Our model contains two main components, a CNN-based autoencoder to learn a latent representation, an transformer-based entropy model to predict latents. We describe our architecture in detail in Figure 2. In this section, we first propose two ingredients, a diamond relative position encoding (diamond RPE) and a top-k scheme, which are essential for image compression. Then, we extend the checkboard context model (He et al., 2021) to a parallel bidirectional context model.

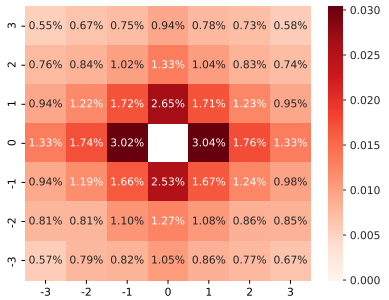

Figure 3: Rate increase ratios of different position when the corresponding context is masked. Closer context contributes to more bitrate influence, whose trend follows a particular pattern of diamond shape.

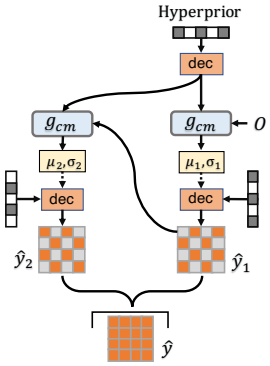

Figure 4: Parallel bidirectional context model. The first slice ($\hat{y}_1$) is conditioned solely on the hyperprior. The second slice ($\hat{y}_2$) is conditioned on the hyperprior and the decoded context (*i.e.* first slice).

3.1 Transformer Architecture

In entropy model design we follow the original Transformer (Vaswani et al., 2017), which employs an encoder-decoder structure. We model the latent representation $\hat{y} \in \mathbb{R}^{H \times W \times C}$ as a sequence $\hat{y}_p \in \mathbb{R}^{(H \cdot W) \times C}$, where (H, W) is the resolution of the latents, C is the number of channels. The latents sequence is then mapped to D dimensions with a trainable linear projection. For hyperprior, we stack N transformer encoder layers to yield hyper-latent z, which is quantized and fed to N transformer-encoder layers to generate hyperprior. To produce a hierarchical representation, the resolution of feature is changed by downsample and upscale module. For autoregressive prior, we stack $2N$ transformer decoder layers to generate autoregressive features. To generate the Gaussian parameters, a linear layer is attached to the combination of hyperprior features and context features.

3.2 Position Encoding

To explore the impact of position in image compression, we first design a transformer-based entropy model without any position encoding. We feed a random mask to self-attention during training. During testing, we evaluate the effect of each position i by employing a corresponding mask, which are set to 1 except position i. Figure 3 plots the impact in bit rate of each position, compared to the result with context of all positions. This result highlights how the rate effected by the position of context. This observation presents a comprehensive understanding and provides empirical guideline for new position encoding design.

In order for the model to make use of the position, we must inject some information about the spacial order of the latents. One common approach is to use biased attention weights based on relative relation (Shaw et al., 2018). Based on the result in Figure 3, we propose an extension to this relation-aware self-attention to consider the element position influence on image compression performance.

Each attention operates on an input sequence, *e.g.*, the embedding of words or the image latents, $y = (y_1, ..., y_n)$ of n elements where $y_i \in \mathbb{R}^{d_y}$. The edge between input elements y_i and y_j is represented by vectors $p_{ij}^K \in \mathbb{R}^{d_k}$. Scaled dot product is chosen for the a compatibility function that compares two input elements in relation-aware self-attention:

$$e_{ij} = \frac{y_i W^Q (y_j W^K + p_{ij}^K)^T}{\sqrt{d_k}} \tag{4}$$

$W^Q, W^K \in \mathbb{R}^{d_y \times d_k}$ are learned parameter matrices. The weight coefficient, A, is computed using a softmax on dot-product attention matrix.

On image plane, the relative position $(a_i - a_j, b_i - b_j)$ is a 2D coordinate. And we modify the maximum absolute relative position to a 2D boundary with diamond shape. As shown in Figure 3, we observe that the closer contexts latent save more bit rate in context model. That tells how the distance of context latents influences the bit saving of context modeling in learned image compression. Therefore, we consider a 2D boundary with a maximum Hamming distance value of h.

$$
\begin{aligned}
p_{ij}^K &= w^K_{clip((a_i-a_j,b_i-b_j),h)} \\
clip\left((a_i - a_j, b_i - b_j), h\right) &= \begin{cases} (a_i - a_j, b_i - b_j) & \|a_i - a_j\| + \|b_i - b_j\| <= h \\ (h, h) & \text{otherwise.} \end{cases}
\end{aligned} \tag{5}
$$

We then learn the relative position encoding $w^K = (w^K_{-h,-h}, w^K_{-h,-h+1}, ..., w^K_{h,h-1}, w^K_{h,h})$.

3.3 Top-k Scheme in Self-Attention

The vanilla attention in the transformer is the scaled dot-product attention. It computes the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values. The queries, keys and values are packed together into matrices $Q \in \mathbb{R}^{n \times d_k}$, $K \in \mathbb{R}^{n \times d_k}$ and $V \in \mathbb{R}^{n \times d_v}$. The matrix of outputs is computed as:

$$
V' = AV = \text{softmax}\left(\frac{Q(K+P)^T}{\sqrt{d_k}}\right) V \tag{6}
$$

where $A \in \mathbb{R}^{n \times n}$ denotes the attention matrix, and $P \in \mathbb{R}^{n \times d_k}$ denotes relative position encodings described in Sec 3.2.

Instead of computing the attention matrix for all the query-key pairs as in vanilla attention, we select the top-k most similar keys for each query. The row-wise top-k elements in attention matrix are selected for softmax function:

$$
V' = \text{softmax}\left(f_k\left(\frac{Q(K+P)^T}{\sqrt{d_k}}\right)\right) V \tag{7}
$$

where $f_k(\cdot)$ denotes the row-wise top-k selection operator:

$$
f_k\left(e_{ij}\right) = \begin{cases} e_{ij} & \text{if } e_{ij} \text{ is within the top-k largest elements in row } j \\ -\infty & \text{otherwise.} \end{cases} \tag{8}
$$

Albeit easing sequence length mismatching problem by relative position encoding, transformer-based entropy model also suffers from token quantity mismatching in self-attention. We suspect that for extremely larger sequence length during inference than training, a large quantity of irrelevant context latents would push the relevant ones to a small weight. To counteract this effect, we modify the self-attention with top-k scheme. It helps to distill irrelevant context latents and stable training procedure.

3.4 Parallel Bidirectional Context Model

The transformer architecture can be used to efficiently train autoregressive model by masking the attention computation such that the i-th position can only be influenced by a position j if and only if $j < i$. This causal masking limits the context and makes autoregressive model impossible to be parallel.

Following the work of He et al. (2021), the latents are split into two slices along the spatial dimension. The first slice ($\hat{y}_1$) is using a Gaussian entropy model conditioned solely on the hyperprior, while the entropy model for the second slice ($\hat{y}_2$) is conditioned on both the hyperprior and the decoded context latents (*i.e.* first slice). Figure 4 provides a high-level overview of this architecture. Since the context are accessible for each latent in second slice, the second slice can be processed in parallel. Therefore, the decoding of this structure can be performed in two serial processing.

When decoding the second slice, we adjust the mask in the self-attention module to build a bidirectional context model, which introduces the future context to benefit accurate prediction. For this reason, though the performance of the first slice degrades (due to hyperprior only), Entroformer

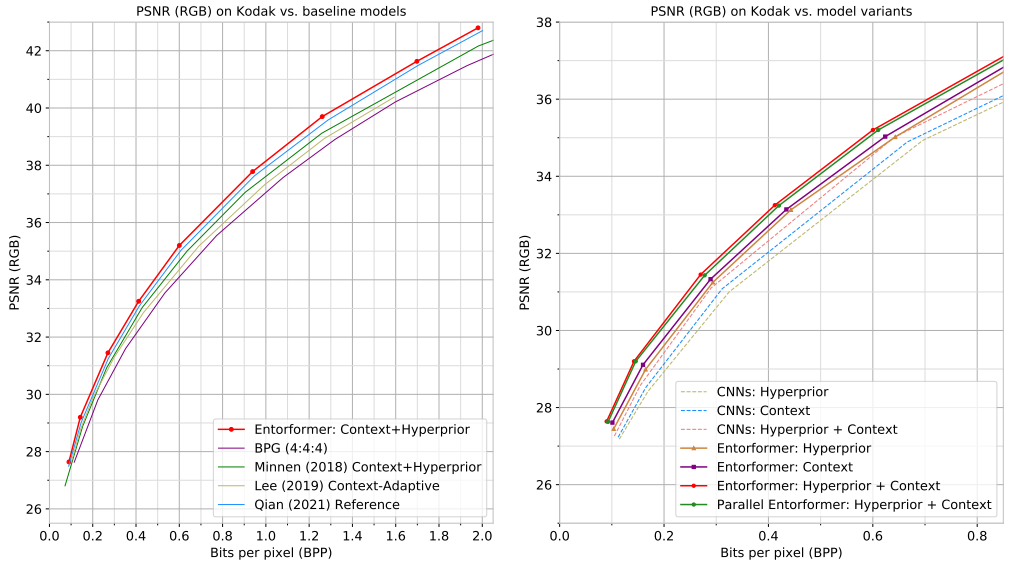

Figure 5: The Entroformer (context + hyperprior) achieves better RD performance than previous CNNs-based methods and standard codecs on the Kodak image set as measured by PSNR (RGB) (left). The right graph compares the relative performance of different versions of our method. Our Entroformer architecture helps significantly at hyperprior-only and context-only model by 12.4% and 11.3% respectively at low bit rates. Note that parallel Entroformer has almost no effect (about 1% bpp increase) for RD performance.

counteracts this effect by providing a more powerful context model to promote the performance of the second slice. In the other hand, compared to the CNN-based accelerated method (about 4% bpp increase) (He et al., 2021), our transformer-based method can utilize rich context and achieve a better balance between speed and performance (about 1% bpp increase).

4 Experimental Results

We evaluate the effects of our transformer-baed entropy model by calculating the rate–distortion (RD) performance. Figure 5 shows the RD curves over the publicly available Kodak dataset (Kodak, 1993) by using peak signal-to-noise ratio (PSNR) as the image quality metric. As shown in the left part, our Entroformer with joint the hyperprior module and the context module outperforms the state-of-the-art CNNs methods by 5.2% and the BPG by 20.5% at low bit rates. As shown in the right part, two variant entropy models, the hyperprior-only one and the context-only one, are evaluated to isolate the effect of Entroformer architecture. Additionally, the parallel Entroformer with bi-directional context is also competitive and is better than the previous baselines.

To evaluate the importance of different components of the Entroformer, we varied our base model in different ways, measuring the change in performance on Kodak test set. When evaluating the effects of using the diamond relative position encoding and the top-k scheme, we adapt the Entroformer with only the context module. When evaluating the effects of bidirectional context model, we employ the Entroformer with the context module and the hyperprior module.

Architecture The architecture of our approach extends on the work of Minnen et al. (2018b). As our focus is on the entropy model, we only replace the CNNs entropy model by proposed Entroformer (*i.e.* the main autoencoder is not changed). We used 6 transformer encoder blocks for hyperprior and 6 transformer decoder blocks for context. The outputs of these two modules are combined by a linear layer. The dimension of the latent embedding is 384. Each transformer layer has a multi-head self-attention (heads=6), and a feed-forward network (ratio=4). If not specified, we use serial Entroformer with the default parameters: k=32 (top-k scheme), h=3 (diamond RPE).

Training Details We choose 14886 images from OpenImage (Krasin et al., 2017) as our training data. During training, all images are randomly cropped to 384 $\times$ 384 patches. Note that large patch

Model	Position Information	Mode	BPP	Δ bpp (%)
CNNs	-	-	0.6621	-
Transformer	-	-	0.6569	0.8
Transformer	Absolute Position Encoding	2D	0.6563	0.9
Transformer	Relative Position Encoding	1D+1D	0.6358	4.0
	Relative Position Encoding	2D	0.6296	4.9
	Relative Position Encoding	2D+Diamond	**0.6238**	**5.8**

Table 1: Ablation of position encoding methods.

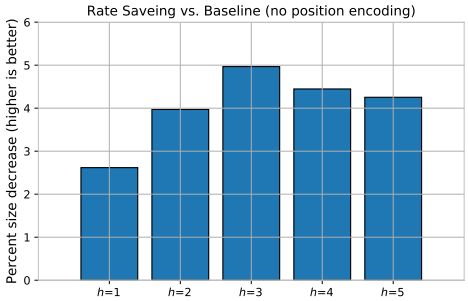

Figure 6: Results for varying the clipping Hamming distance of diamond RPE. The base implementation uses a transformer context model without position encoding. All models are optimized with $\lambda = 0.02$ (PNSR $\approx$ 35.0 on Kodak).

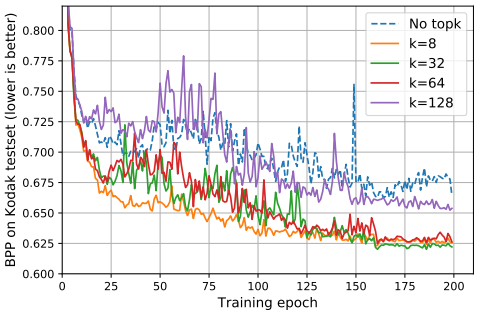

Figure 7: Effect of restricting the self-attention to various top-k similar content. All models uses a transformer context model and are optimized with $\lambda = 0.02$ (PNSR $\approx$ 35.0 on Kodak).

size is necessary for the training of the transformer-based model. We optimize the networks using distortion terms of MSE. For each distortion type, the average bits per pixel (BPP) and the distortion PSNR, over the test set are measured for each model configurations. The detailed structure and the experimental settings are described in appendix.

4.1 Analysis on Position Encoding

Position Encoding Methods We perform a component-wise analysis to study the effects of different position encodings in our Entroformer. As shown in Table 1, When applying absolute position encoding or non-position encoding in Entroformer, we can achieve a limited bpp saving than the baseline. If we apply the relative position encoding, it can perform better than the absolute position one (4.0% bpp saving *vs*. 0.9% bpp saving). Additionally, it is crucial to extend the position encoding from 1D to 2D (4.9% bpp saving *vs*. 4.0% bpp saving) . Finally, combining with the diamond boundary, we could get more bit-rate saving than the others, in particular, 5.8% bpp saved than the CNNs one.

Clipping Distance We evaluate the effect of varying the clipping Hamming distance, h, of diamond RPE. Figure 6 compares the number of h by showing the relative reduction in bit rate at a single rate-point. The baseline is implemented by a transformer-based context model without position encoding. It achieves the best result when $h = 3$. The conclusion is similar as the analysis shown in Figure 3.

4.2 Number k of top-k self-attention

The impact of top-k scheme is analyzed in Figure 7. The parameter k specifies the involved token number of self-attention. We report the full learning bpp curves on Kodak test set for different number k and not just single RD point. Additional curve of original self-attention mechanism is presented (dash) for comparison. The spatial size of latents is 24×24 during training and 32×48 during testing. Notably, the compression performance increase for $k \leq 64$, which is much

Test set	Ballé2018 (Hyperprior)	Minne2018 (Hyperprior+Context)	Entroformer (Hyperprior+Context)	
			Serial	parallel
Kodak (768 × 512)	21.6	4229.9	51678.5	170.6

Table 2: Inference latency (ms) of entropy model on Kodak. Ballé et al. (2018) uses a entropy model conditioned on hyperprior. Minnen et al. (2018b) uses a context-adaptive entropy model conditioning on the causal context, which is inherently serial. For a direct comparison, we test our Entroformer with serial architecture and parallel architecture.

Half latents ($\hat{y}_2$)	Hyperprior-only	Hyperprior & unidirectional context	Hyperprior & bidirectional context
BPP	0.3267	0.3105	0.2972

Table 3: Bidirectional context *vs.* unidirectional context. Combined with bidirectional context, context model achieves bit rate saving significantly. The results are test by the same model.

smaller than the sequance length of 576 and 1536 for training and testing respectively. Moreover, the top-k scheme also influences convergence speed of Entroformer. Interestingly, when increase k larger than 64, we see quite different results that top-k scheme has no difference with original self-attention. One hypothesis is that there is a vast amount of irrelevant information in dense self-attention for image compression. These observations reflect that removing the irrelevant tokens benefits the convergence of Entroformer training.

4.3 Parallel Bidirectional Context Model

In Table 2, we report the inference latency of Entroformer (serial architecture and parallel architecture) comparing to other learned entropy models. We benchmark models' inference latency on 16GB Tesla V100 GPU card. From Table 2, we see that parallel bidirectional context model is much more time-efficient than serial unidirectional context model.

As shown in Figure 5, by employing Entroformer, performance of the parallel bidirectional context is on par with the serial unidirectional context. For checkboard parallel context model (He et al., 2021), it speeds up the decoding process with a loss of compression quality. We also study the impact of bidirectional context for compression performance on the second slice (*i.e.* $\hat{y}_2$) solely. Table 3 shows that bidirectional context has more accurate prediction than unidirectional context in our Entroformer. This experiment explains the ability of the parallel Entroformer with bidirectional context.

4.4 Visualization

Attention mechanism in transformer is the key component which models relations between feature representations. In this section, we visualize the self-attentions mechanism of our Entroformer, focusing on a few points in the image. The attention map is rescaled for a better view. In Figure 8, we visualize self-attention heads from different points These attention heads exhibit different behaviours that seem related to structure, semantic and color respectively. These visualizations show how the Entroformer finds related context to support its distribution prediction for the current latent. This allows the Entroformer to capture richer dependencies of the latents and achieve better compression performance. In Figure 9, we visualize self-attentions heads separately from one point. We can see that different heads attend to different region of an image. This means that separate sections of the representation can learn different aspects of the meanings of each latent. In Figure 10, we visualize self-attentions heads with different k. Compared with dense attention, the top-k attention filters out irrelevant information from background. It successfully makes the attention concentrating on the most informative context.

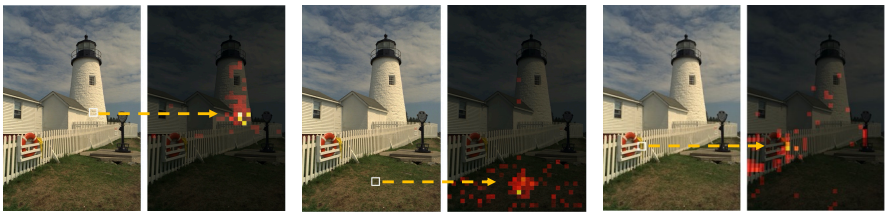

Figure 8: Examples of the self-attention mechanism for three points.

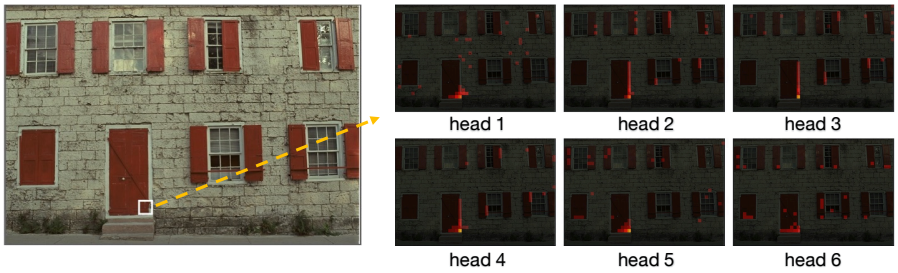

Figure 9: Self-attention heads following long-range dependencies with different patterns.

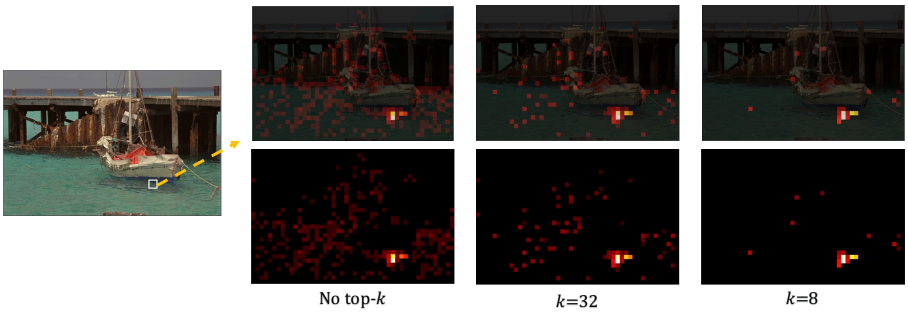

Figure 10: Self-attention from models with different k.

5 Discussion

In this paper, Entroformer is proposed as the entropy model to improve learned image compression by accuracy probability distribution estimation. To our knowledge, this is the first successful attempt to introduce transformers in image compression task. The Entroformer follows the standard transformer framework with a novel positional encoding and top-k self-attention module. The proposed diamond RPE is a 2D relative position encoding with a diamond shape boundary. The top-k self-attention is helpful in dependencies distillation. In particular, the top-k module benefits the convergence of neural networks training. Last but not least, the Entroformer with bidirectional context can work well in the two-pass decoding framework for fast speed without performance degradation. Experiments show that the Entroformer achieves the SOTA results in standard benchmarks.

There are still some issues to be discussed in the future. For example, (1) Whether we could build a hierarchical decoding framework to achieve a better balance between speed and RD performance, even beyond the raster-scan mode; (2) Whether the fine-grained assignment of k to different images could increase performance; (3) Whether the rate control module plays a more important role in the transformer-based entropy model than in the convolution-based model. We would like to explore these issues in future work.

REFERENCES

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimization of nonlinear transform codes for perceptual quality. In *Picture Coding Symposium*, pp. 1–5. IEEE, 2016.

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *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. In *International Conference on Learning Representations*, 2018.

Fabrice Bellard. Bpg image format, 2014. URL `http://bellard.org/bpg/`.

Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7939–7948, 2020.

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*, pp. 14771–14780, 2021.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Yueyu Hu, Wenhan Yang, and Jiaying Liu. Coarse-to-fine hyper-prior modeling for learned image compression. In *AAAI*, pp. 11013–11020, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Eastman Kodak. Kodak lossless true color image suite (photocd pcd0992), 1993. URL `http://r0k.us/graphics/kodak/`.

Ivan Krasin, Tom Duerig, Neil Alldrin, Vittorio Ferrari, Sami Abu-El-Haija, Alina Kuznetsova, Hassan Rom, Jasper Uijlings, Stefan Popov, Shahab Kamali, Matteo Malloci, Jordi Pont-Tuset, Andreas Veit, Serge Belongie, Victor Gomes, Abhinav Gupta, Chen Sun, Gal Chechik, David Cai, Zheyun Feng, Dhyanesh Narayanan, and Kevin Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://storage.googleapis.com/openimages/web/index.html*, 2017.

Jooyoung Lee, Seunghyun Cho, and Seung-Kwon Beack. Context-adaptive entropy model for end-to-end optimized image compression. *International Conference on Learning Representations*, 2019.

Mu Li, Wangmeng Zuo, Shuhang Gu, Debin Zhao, and David Zhang. Learning convolutional networks for content-weighted image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3214–3223, 2018.

Mu Li, Kai Zhang, Wangmeng Zuo, Radu Timofte, and David Zhang. Learning context-based non-local entropy modeling for image compression. *arXiv preprint arXiv:2005.04661*, 2020.

Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4394–4402, 2018.

David Minnen and Saurabh Singh. Channel-wise autoregressive entropy models for learned image compression. In *2020 IEEE International Conference on Image Processing*, pp. 3339–3343. IEEE, 2020.

David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. In *Advances in Neural Information Processing Systems*, pp. 10771–10780, 2018a.

David Minnen, George Toderici, Saurabh Singh, Sung Jin Hwang, and Michele Covell. Image-dependent local entropy models for image compression with deep networks. 2018b.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32: 8026–8037, 2019.

Yichen Qian, Zhiyu Tan, Xiuyu Sun, Ming Lin, Dongyang Li, Zhenhong Sun, Hao Li, and Rong Jin. Learning accurate entropy model with global reference for image compression. *International Conference on Learning Representations*, 2021.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. In *International Conference on Learning Representations*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

A APPENDIX

A.1 ARCHITECTURE DETAILS

Encoder	Decoder	Hyper Encoder	Hyper Decoder	Context Model	Linear
Conv: k5c192s2	Deconv: k5c192s2	Trans: dim384head6	Trans: dim384head6	Trans: dim384head6	Linear: dim768
GDN	IGDN	Downscale	Upscale	Trans: dim384head6	Leaky ReLU
Conv: k5c192s2	Deconv: k5c192s2	Trans: dim384head6	Trans: dim384head6	Trans: dim384head6	Linear: dim768
GDN	IGDN	Downscale	Upscale	Trans: dim384head6	
Conv: k5c192s2	Deconv: k5c192s2	Trans: dim384head6	Trans: dim384head6	Trans: dim384head6	
GDN	IGDN			Trans: dim384head6	
Conv: k5c384s2	Deconv: k5c192s2				

Table 4: Each row corresponds to a layer of our generalized model. Convolutional layers are specified with the "Conv" prefix followed by the kernel size, number of output channels and downsampling stride (*e.g*. the first layer of the encoder uses 5×5 kernels with 192 output channels and a stride of 2). The "Deconv" prefix corresponds to upsampled convolutions. The "Trans" corresponds to transformer block as in Vaswani et al. (2017), followed by the inner dimension and head number. A transformer block consists of a self-attention module and a feed-forward module (the dimension of the feed-forward is 4 times of inner embedding's). GDN stands for generalized and divisive normalization, and IGDN is inverse GDN (Ballé et al., 2016).

Details about the individual network layers in each component of our models are outlined in Table 4. The output of the last layer in encoder corresponds to the latents $\hat{y}$, and the output of the last layer in hyper-encoder corresponds to the hyper-latents $\hat{z}$. The output of the last layer in decoder corresponds to the generated RGB image $\hat{x}$. The hyperprior feature and context feature are concatenated and fed into two "Linear" layers to yield the parameter, mean and deviation of a Gaussian distribution ($\mathcal{N}(\mu, \sigma^2)$), for each latent. The output of the linear network must have exactly twice as many channels as the latents.

A.2 TRAINING DETAILS

We implement all our models with PyTorch(Paszke et al., 2019). We use the Adam optimizer (Kingma & Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-8$, and base learning rate=$3e-5$. When training transformers, it is standard practice to use a warmup phase at the beginning of learning, during which the learning rate increases from zero to its peak value Vaswani et al. (2017). We use a warmup with 0.05 proportion of the total epochs. And then the learning rate decays stepwise for every 100 epochs by a factor of 0.3. Gradient clipping is also helpful in the compression setup, which is set to 1.0. No weight decay is used in our models because rate constrain is inherently a regularization. We initialize the weights of our Entroformer with a truncated normal distribution.

All models are trained for 300 epochs with a batchsize of 16 and a patch size of 384×384 on 16GB Tesla V100 GPU card. To obtain models for different bit rate, we adapt the hyper-parameter λ to cover a range of rate-distortion tradeoffs.