Efficient and Effective Context-Based Convolutional Entropy Modeling for Image Compression

Mu Li, Kede Ma, Member, IEEE, Jane You, Member, IEEE, David Zhang, Fellow, IEEE, and Wangmeng Zuo, Senior Member, IEEE

Abstract—It has long been understood that precisely estimating the probabilistic structure of natural visual images is crucial for image compression. Despite remarkable success of recent end-to-end optimized image compression, the latent code representation is assumed to be fully statistically factorized such that the entropy modeling is feasible. Here we describe context-based convolutional networks (CCNs) that exploit statistical redundancies in the codes for improved entropy modeling. We introduce a 3D zigzag coding order together with a 3D code dividing technique to define proper context and to achieve parallel entropy decoding, both of which boil down to place translation-invariant binary masks on convolution filters of CCNs. We demonstrate the power of CCNs for entropy modeling in both lossless and lossy image compression. For the former, we directly apply a CCN to binarized image planes for estimating the Bernoulli distribution. For the latter, we propose a novel, highly efficient approach that jointly optimizes the CCN-based entropy model with analysis distributions, whose parameters are estimated by three CCNs. Experiments on two image datasets show that the proposed lossless and lossy image compression methods based on CCNs generally exhibit better compression performance than existing methods with a manageable computational complexity.

Index Terms—Context-based convolutional networks, entropy modeling, image compression.

I. INTRODUCTION

Data compression has played a significant role in engineering for centuries [1]. Compression can be either lossless or lossy. Lossless compression allows perfect data reconstruction from compressed bitstreams with the goal of assigning shorter codewords to more “probable” codes. Typical examples include Huffman coding [2], arithmetic coding [3], and range coding [4]. Lossy compression discards “unimportant” information, and the definition of importance is application-dependent. For example, if the data (such as images and videos) are meant to be consumed by the human visual system, importance should be measured in accordance with human perception, discarding data that are perceptually redundant, while keeping those that are most visually noticeable. In lossy compression, one must face the rate-distortion trade-off, where the rate is computed by the entropy of the discrete codes [5] and the distortion is measured by a signal fidelity metric. A prevailing scheme in the context of lossy image compression is transform coding, which consists of three operations - transformation, quantization, and entropy coding. Transforms are designed to be better-suited for exploiting aspects of human perception, and map an image to a latent code space with several statistical advantages. Early transforms [6], [7] are linear, invertible, and fixed for all bit rates; errors arise only from quantization. Recent transforms take the form of deep neural networks (DNNs) [8], aiming for more comprehensible and nonlinear representations. DNN-based transforms are mostly non-invertible, which, however, may encourage discarding perceptually unimportant image features during transformation. This gives us an opportunity to learn different transforms at different bit rates for better rate-distortion performance. Entropy coding is responsible for losslessly compressing the quantized codes into bitstreams for storage and transmission.

In either lossless or lossy image compression, a discrete probability distribution of the latent codes shared by the encoder and the decoder (i.e., the entropy model) is essential in determining the compression performance. According to Shannon’s source coding theorem [5], given a vector of code intensities \( y = \{y_0, \ldots, y_M\} \), the optimal code length of \( y \) should be \( \lceil -\log_2 P(y) \rceil \), where binary symbols are assumed to construct the codebook. Without further constraints, the general problem of estimating \( P(y) \) in high-dimensional spaces is intractable, a problem commonly known as the curse of dimensionality. For this reason, most entropy coding schemes assume \( y \) is fully statistically factorized with the same marginal distribution, leading to a code length of \( \lceil -\sum_{i=0}^{M} \log_2 P(y_i) \rceil \). Alternatively, the chain rule in probability theory offers a more accurate approximation

\[
P(y) \approx \prod_{i=0}^{M} P(y_i | PTX(y_i, y)),
\]

where \( PTX(y_i, y) \subset \{y_0, \ldots, y_{i-1}\} \) represents the partial context of \( y_i \) coded before it in \( y \). A representative example is the context-based adaptive binary arithmetic coding (CABAC) [9] in H.264/AVC, which considers the two nearest codes as the context and obtains noticeable improvements over previous image/video compression standards. As the size of \( PTX(y_i, y) \) becomes large, it is difficult to estimate this conditional probability by constructing histograms. Recent
methods such as PixelRNN [10] and PixelCNN [11] take advantage of DNNs in modeling long range relations among pixels, making the estimation of $P(y_i | P\text{TX}(y_i, y))$ with larger partial contexts feasible. Nevertheless, these methods are computationally intensive, taking hours for medium-resolution images.

We describe context-based convolutional networks (CCNs) for parallel entropy modeling. Given $y$, we specify a 3D zigzag coding order so that the most relevant codes of $y_i$ can be included in its context. Parallel computation during entropy encoding is straightforward as the context of each code is known and readily available. However, this is not always the case during entropy decoding because the context of $y_i$ is not ready for probability estimation until all codes in $P\text{TX}(y_i, y)$ have been decoded sequentially, which is prohibitively slow. To address this issue, we introduce a 3D code dividing technique, which partitions $y$ into multiple groups in compliance with the proposed coding order. The codes within each group are assumed to be conditionally independent given their respective contexts, and therefore can be decoded in parallel. In the context of CCNs, this amounts to applying properly designed translation-invariant masks to convolutional filters.

To validate the proposed CCNs, we combine them with arithmetic coding [3] for entropy coding. For lossless image compression, we convert the input grayscale image into eight binary planes and train a CCN to predict the Bernoulli distribution of $y_i$, optimizing for the loss error in information theory [12]. For lossy image compression, we parameterize the the categorical distribution of $y_i$ with a discretized mixture of Gaussian (MoG) distributions, whose parameters (i.e., mixture weights, means, and variances) are estimated by three CCNs, depending on its context. The CCN-based entropy model is jointly optimized with analysis and synthesis transforms (i.e., mappings between raw pixel space and latent code space) over a database of training images for rate-distortion performance. Across two independent sets of images, we find that the optimized methods for lossless and lossy image compression perform favorably against image compression standards and DNN-based methods, especially at low bit rates.

II. RELATED WORK

In this section, we provide a detailed overview of DNN-based entropy models and lossy image compression methods. For traditional image compression techniques with separately optimized components, we refer the interested readers to [13], [14], [15].

A. DNN-Based Entropy Models

The accuracy of entropy coding largely affects rate-distortion performance of image compression systems. In this process, the first and the most important step is to estimate the probability $P(y)$. In most image compression techniques, $y$ is assumed to be fully statistically factorized, whose entropy can be easily computed through the marginal distributions [8], [16], [17], [18]. Arguably speaking, natural images that undergo a highly nonlinear analysis transform still exhibit strong statistical redundancies [19], which suggests that incorporating context into probability estimation has great potentials in improving entropy modeling performance.

Considerable progress has been made in DNN-based context modeling for natural languages and images. In natural language processing, recurrent neural networks (RNN) [20], and long short-term memories [21] are two popular tools to model long-range dependencies. In image processing, PixelRNN [10] and PixelCNN [11] are among the first attempts to exploit long-range pixel dependencies for image generation. However, the above-mentioned methods are computational inefficient, which require one forward propagation to generate a single pixel or to estimate its probability. To speed up PixelCNN, Salimans et al. [22] proposed PixelCNN++, which is able to sample a larger intermediate image conditioned on an initial image in a single forward pass. This process may be iterated to generate the final result. When viewing PixelCNN++ and its variants as entropy models, we must losslessly compress the initial image into bitstreams as side information, and send them to the decoder for entropy decoding.

Only recently has DNNs for context-based entropy modeling become an active research topic. Ballé et al. [19] introduced a scale prior, which stores a variance parameter for each $y_i$ as side information. Richer side information generally leads to more accurate entropy modeling. However, as discussed before, this type of information should be quantized, compressed and considered part of the codes, and it is difficult to trade off the bits saved by the improved entropy model and the bits introduced by storing this information. Li et al. [25] extracted a small code block for each $y_i$ as its context, and adopted a simple DNN for entropy modeling. The method suffers from heavy computational complexity similar to PixelRNN [10]. Li et al. [24] and Mentzer et al. [25] achieved parallel entropy encoding with mask DNNs. However, sequential entropy decoding has to be performed due to the context dependence of the codes, which is painfully slow. In contrast, our CCN-based entropy model equipped with a 3D zigzag coding order and a 3D code dividing technique permits parallel entropy coding, and is a more practical choice for industrial usage.

B. DNN-Based Lossy Image Compression Methods

A major problem in end-to-end lossy image compression is that the gradients of the quantization function are zeros almost everywhere, making gradient descent-based optimization ineffective. Based on the strategies of alleviating the zero-gradient problem of quantization, DNN-based lossy image compression methods can be divided into different categories.

From a signal processing perspective, the quantizer can be approximated with an additive i.i.d. uniform noise, which has the same width as the quantization bin [26]. A desired property of this approximation is that the resulting density function is a continuous relaxation of the probability mass function of $y$ [8]. Another line of research introduced more continuous functions (without the zero-gradient problem) to approximate the quantization function. The step quantizer is used in the forward pass, while its continuous proxy is used in the backward pass. Toderici et al. [27] learned an
RNN to compress small-size images to binary maps in a progressive manner. They later tested their models on large-size images [28]. Johnston et al. [29] exploited adaptive bit allocations and perceptual losses to boost the compression performance especially in terms of MS-SSIM [30]. These methods are not explicitly optimized for rate-distortion performance, but treat entropy coding of the binary codes as a post-processing step. Ballé et al. [8] explicitly formulated DNN-based image compression under the framework of rate-distortion optimization. Assuming \( y \) is statistically factorized, they learned piece-wise linear density functions to compute differential entropy as approximation to discrete entropy. In a subsequent work [19], each \( y_t \) is assumed to zero-mean Gaussian with its own standard deviation, which is separately predicted with side information. Minnen et al. [31] combined the autoregressive and hierarchical priors, which lead to improved rate-distortion performance. Theis et al. [16] introduced a continuous upper bound of the discrete entropy with a Gaussian scale mixture. Rippel et al. [32] described pyramid-based analysis and synthesis transforms with adaptive code length regularization for real-time image compression. An adversarial loss [32] is incorporated to generate visually realistic results at low bit rates [18]. Agustsson et al. [17] introduced a soft-to-hard relaxation scheme by approximating quantization with a parametric softmax function.

III. CCNs FOR ENTROPY MODELING

In this section, we present in detail the construction of CCNs for entropy modeling. We start with a fully convolutional network, consisting of \( T \) layers of convolutions followed by point-wise nonlinear activation functions. In order to perform efficient context-based entropy modeling, three assumptions are made on the network architecture:

- For a code block \( y \in \mathbb{R}^{M \times H \times W} \), where \( M, H, \) and \( W \) denote channel, height, and width, respectively, the output of the \( t \)th layer convolution \( v^{(t)} \in \mathbb{R}^{M \times H \times W \times N_t} \), where \( N_t \) denotes the number of feature blocks to represent \( y \). By doing so, we are able to associate the feature point \( v_{i,j}^{(t)} \) in \( i \)th channel and \( j \)th feature block at spatial location \( (p,q) \) with \( y_{i,j}(p,q) \).

- Let \( \text{CTX}(y_{i,j}(p,q), y) \) be the set of codes encoded before \( y_{i,j}(p,q) \) (full context), and \( \text{SS}(v_{i,j}^{(t)}(p,q)) \) be the set of codes in the receptive field of \( v_{i,j}^{(t)}(p,q) \) that contributes to its computation (support set), respectively. Then, \( \text{SS}(v_{i,j}^{(t)}(p,q)) \subset \text{CTX}(y_{i,j}(p,q), y) \).

- For \( y' \subset y \) and \( y_{i,j}(p,q) \in y' \), \( \text{CTX}(y_{i,j}(p,q), y') \subset \text{CTX}(y_{i,j}(p,q), y) \).

Assumption I establishes a one-to-many correspondence between the input code block \( y \) and the output feature representation \( v^{(T)} \). Assumption II ensures that the computation of \( v_{i,j}^{(t)}(p,q) \) depends only on a subset of \( \text{CTX}(y_{i,j}(p,q), y) \). Together, the two assumptions guarantee the legitimacy of context-based entropy modeling in fully convolutional networks, which can be achieved by placing binary masks to convolution filters. Then, \( P(y_{i,j}(p,q)|\text{CTX}(y_{i,j}(p,q), y)) \) can be computed from \( \{v_{i,j}^{(T)}(p,q)\} \), which specify the parameters of the conditional distribution. Assumption III allows the masks to be translation-invariant, which can be achieved by properly designed coding orders. Specifically, we start with the case of a 2D code block, where \( y \in \mathbb{R}^{H \times W} \), and write mask convolution at the \( t \)th layer as

\[
v_{i}^{(t)}(p,q) = \sum_{j=1}^{N_t} (u_{i}^{(t)} * (m_{i}^{(t)} \odot w_{i,j}^{(t)}))(p,q) + b_{i}^{(t)},
\]

where \( * \) and \( \odot \) denote 2D convolution and Hadamard product, respectively. \( u_{i}^{(t)} \) is a 2D convolution filter and \( m_{i}^{(t)} \) is the corresponding 2D binary mask. According to Assumption I, the input \( u_{i}^{(t)} \) and the output \( v_{i}^{(t)} \) are of the same size as \( y \).

The input code block \( y \) corresponds to \( u_{0}^{(0)} \).

For the input layer of a fully convolutional network, the codes to produce \( v_{0}^{(0)}(p,q) \) is \( \Omega_{p,q} = \{y(p+u,q+v)\}_{(u,v) \in \Omega} \), where \( \Omega \) is the set of local indexes centered at \((0,0)\). We choose

\[
\text{SS}(v_{0}^{(0)}(p,q)) = \text{CTX}(y(p,q), \Omega_{p,q}) \subset \text{CTX}(y(p,q), y),
\]

which can be achieved by setting

\[
m_{0}(u,v) = \begin{cases} 1, & \text{if } (u+v) \in \text{CTX}(y(p,q), y) \\ 0, & \text{otherwise} \end{cases}
\]

which can be achieved by setting

\[
m_{0}(u,v) = \begin{cases} 1, & \text{if } (u+v) \in \text{CTX}(y(p,q), y) \\ 0, & \text{otherwise} \end{cases}
\]

where \( \text{CTX}(y(p,q), y) \) is the set of local indexes centered at \((0,0)\). We choose

\[
\text{SS}(u_{j}^{(t)}(p+u,q+v)) \subset \text{CTX}(y(p+u,q+v), y) \subset \text{CTX}(y(p,q), y),
\]

Fig. 1 illustrates the concepts of full context \( \text{CTX}(y(p,q), y) \), support set \( \text{SS}(v_{j}^{(0)}(p,q)) \), and translation-invariant mask \( m_{0}(0) \), respectively. At the \( t \)th layer, if a code \( y(p+u,q+v) \in \text{CTX}(y(p,q), y) \), we have

\[
\text{SS}(u_{j}^{(t)}(p+u,q+v)) \subset \text{CTX}(y(p+u,q+v), y) \subset \text{CTX}(y(p,q), y),
\]

where the first line follows by induction and the second line follows from the definition of context. That is, as long as \( y(p+u,q+v) \) is in the context of \( y(p,q) \), we are able to compute \( v_{i}^{(t)}(p,q) \) from \( u_{j}^{(t)}(p+u,q+v) \) without violating Assumption II. Unlike the input layer, \( u_{j}^{(t)}(p,q) \) with \( t > 0 \) is also generated from \( \text{CTX}(y(p,q), y) \), and can be used to
Fig. 2. Illustration of code dividing techniques in conjunction with different coding orders for a 2D code block. The orange and blue dots represent two nearby codes. The gray dots denote codes that have already been encoded, while the white circles represent codes yet to be encoded. (a) Raster coding order adopted in many compression methods. (b) Support sets of the orange and blue codes, respectively. It is clear that the orange code is in the support set of the blue one, and therefore should be decoded first. (c) Code dividing scheme for the raster coding order. By removing the dependencies among codes in each row, the orange and blue codes can be decoded in parallel. However, the orange code is excluded from the support set of the blue one, which may hinder entropy estimation accuracy. (d) Zigzag coding order and its corresponding code dividing scheme. The two codes in the orange squares that are important for the orange code in entropy prediction are retained in its partial context. (e) Support sets of the orange and blue codes in compliance with the zigzag coding order.

With the proposed CCN with translation-invariant masks in Eq. (4) and Eq. (6), \( y \) can be encoded in parallel. However, this is not the case in the entropy decoding phase. As shown in Fig. 2 (a) and (b), the two nearby codes in the same row (highlighted in orange and blue, respectively) could not be decoded simultaneously because the orange code is in the support set (or context) of the blue code given the raster coding order. To speed up entropy decoding with parallel computation, we may further remove dependencies among codes at the risk of model accuracy. Specifically, we partition \( y \) into \( K \) groups, namely, \( \text{GP}_0(y) = \{ y_{r(p,q)} \} \) and \( \text{GP}_{k-1}(y) \), and remove dependencies among the codes within the same group, resulting a partial context \( \text{PTX}(y_{(p,q)}) = \{ \text{GP}_0(y), \ldots, \text{GP}_{k-1}(y) \} \) for \( y_{(p,q)} \in \text{GP}_k(y) \). As a result, all codes in the \( k \)th group share the same partial context, and can be decoded in parallel. Note that code dividing schemes are largely constrained by the pre-specified coding order. For example, if we use a raster coding order, it is straightforward to divide \( y \) by row. In this case, \( y_{(p,q)} \) according to CABAC [9], has been excluded from its

$$m^{(t)}(u,v) = \begin{cases} m^{(0)}(u,v), & \text{if } (u,v) \neq (0,0) \\ 1, & \text{otherwise.} \end{cases}$$  

(6)
The proposed CCN-based entropy model for lossless image compression. The grayscale image \( x \) is first converted to bit-plane representation \( y \), which is fed to the network to predict the mean estimates of Bernoulli distributions \( P[y_{r}(p,q)|SS(v_{r}(p,q))] \). The size of convolution filters and the number of feature blocks in intermediate layers are set to \( S \times S \) and \( N \), respectively. Each convolution layer is followed by a parametric ReLU nonlinearity, except for the last layer, where a sigmoid function is applied. From the mean estimates, we find that for most significant bit-planes, our model makes more confident predictions closely approaching local image structures. For least significant bit-planes, our model is less confident, producing mean estimates close to 0.5.

**IV. CCN-BASED ENTROPY MODELS FOR LOSSLESS IMAGE COMPRESSION**

In this section, we combine our CCN-based entropy model with the arithmetic coding algorithm for lossless image compression.

As a starting point, we binarize the grayscale image \( x \in \mathbb{R}^{H \times W} \) to obtain a 3D code block

\[
y_{r}(p,q) = \left\lfloor \frac{x(p,q)}{2^{2T-r}} \right\rfloor \mod 2, \quad r = 0, 1, \ldots, 7, \tag{11}
\]

where we index the most significant bit-plane with \( r = 0 \). Our CCN takes \( y \) as input and produces a feature block \( v \) (the superscript \( T \) is omitted for notation convenience) of the same size to compute the mean estimates of Bernoulli distributions \( P(y_{r}(p,q)|SS(v_{r}(p,q))) \). Fig. 5 shows the network architecture, which consists of eleven mask convolution layers with parametric ReLU nonlinearities in between. The last convolution responses undergo a sigmoid nonlinearity to constrain the dynamic range within [0, 1]. We make four residual connections as suggested in [33] to accelerate training. We will experiment with two hyper-parameters in CCN: the size of convolution filters for all layers \( S \) and the number of feature blocks in hidden layers \( N \).

To optimize the network parameters, which are collectively denoted by \( \theta \), we apply the expected code length as the empirical loss

\[
\ell(\theta) = \mathbb{E}_{y} \left[ -\sum_{r,p,q} \left( \mathbb{I}(y_{r}(p,q) = 1) \log_{2}(v_{r}(p,q)) + \mathbb{I}(y_{r}(p,q) = 0) \log_{2}(1 - v_{r}(p,q)) \right) \right], \tag{12}
\]

where \( \mathbb{I}(\cdot) \) is an indicator function and the expectation may be approximated by averaging over a mini-batch of training images. Finally, we implement our own arithmetic coding with the learned CCN-based entropy model to compress \( y \) to bitstreams, and report performance using actual bit rates. This facilitates comparison against widely used image compression standards.

**V. CCN-BASED ENTROPY MODELS FOR LOSSY IMAGE COMPRESSION**

In lossy image compression, our objective is to minimize a weighted sum of rate and distortion, \( \ell_{r} + \lambda \ell_{d} \), where \( \lambda \)
Down-sampling: implemented jointly with the adjacent convolution (also referred to as stride convolution). DenseBlock: model, and a synthesis transform. Empirically, the parameter $M$ takes a color image as input and produces the latent code by mapping the parameter vector $\theta$ to its step quantization function. During training, we use $g_d$ and $\hat{g}_d$ in the forward and backward passes, respectively.

The quantization centers $\omega$ should be optimized by minimizing the mean squared error (MSE),

$$\ell_q(\omega) = \frac{1}{MHW} \sum_{r,p,q} \| y_r(p,q) - \tilde{y}_r(p,q) \|_2^2,$$

which is essentially a $k$-means clustering problem, and can be solved efficiently by the Lloyd’s algorithm [39]. Specifically, we initialize $\omega$ using uniform quantization, which appears to work well in all experiments. To make parameter learning of the entire model smoother, we adjust $\omega$ using stochastic gradient descent instead of a closed-form update.

Without prior knowledge of the categorical distributions of the quantized codes $\tilde{y}$, we choose to work with discretized MoG distributions, whose parameters are predicted by the proposed CCNs. We write the differentiable MoG distribution with $C$ components as

$$\tilde{y}_r(p,q) \sim \sum_{i=0}^{C-1} \pi_i \mathcal{N}(\tilde{y}_r(p,q); \mu_i, \sigma^2_i),$$

where $\pi_i$, $\mu_i$, and $\sigma^2_i$ are the mixture weight, mean, and variance of the $i$-th component, respectively. Then,

$$P(\tilde{y}_r(p,q)) = \int_{\Delta} \sum_{i=0}^{C-1} \pi_i \mathcal{N}(\xi; \mu_i, \sigma^2_i) d\xi,$$

where $\Delta$ is the quantization bin that $\tilde{y}_r(p,q)$ lies in.

Now we describe the proposed entropy model in lossy image compression, which is comprised of three CCNs with the same structure, as shown in Fig. 6. Each CCN consists of nine mask convolutions with three residual connections to produce $C$ feature blocks, matching the number of components in MoG. They separately output mixture weights, means and variances to build the discretized MoG distributions. The network parameters of our CCNs constitute the parameter vector $\theta$ to be optimized.
Now we are able to write the empirical rate-distortion objective for the parameters $\{\phi, \psi, \theta\}$ as

$$\ell(\phi, \psi, \theta) = \mathbb{E}_x \left[ - \sum_i \log_2 P_{\hat{y}_i}(g_d(g_a(x; \phi); \psi); \theta) \right] + \lambda_d \left( g_s(g_d(g_a(x; \phi); \psi); \psi); x \right).$$  \hspace{1cm} (17)

$\ell_d$ is the distortion term, which is more preferable to be assessed in a perceptual space. In this paper, we optimize and evaluate our lossy image compression methods using standard MSE and a perceptual metric MS-SSIM [30]. Similar in lossless image compression, we combine the optimized entropy model with arithmetic coding, and measure the rate using actual bit rates.

VI. EXPERIMENTS

In this section, we test the proposed CCN-based entropy models in lossless and lossy image by comparing to state-of-the-art image coding standards and recent deep image compression algorithms. We first collect 10,000 high-quality and high-definition images from Flickr, and down-sample them to further reduce possibly visible artifacts. We crop 1,280,000 grayscale patches of size $128 \times 128$ and 640,000 color patches of size $3 \times 256 \times 256$ as the training sets for lossless and lossy image compression, respectively. We test our models on two independent datasets - Kodak and Tecnick [40], which are widely used to benchmark compression performance. The Caffe implementations along with the pre-trained models are made available at https://github.com/limuhit/SCAE.

A. Lossless Image Compression

We train our CCN-based entropy model using the Adam stochastic optimization package [41] by minimizing the objective in Eq. (12). We start with a learning rate of $10^{-4}$, and subsequently lower it by a factor of 10 when the loss plateaus, until $10^{-6}$. The (actual) bit rate in terms of bits per pixel (bpp) is used to quantify the compression performance, which is defined as the ratio between the total amount of bits used to code the image and the number of pixels in the image. A smaller bpp indicates better performance. For example, an uncompressed grayscale image has eight bpp.

We first compare the proposed CCNs with mask convolutional networks (MCNs) [24], [25], PixelCNN++ [22], and side information networks (SINs) [19] for entropy modeling. As a special case of CCNs, MCNs specify the raster coding order without using any code dividing technique (see Fig. 2). We implement our own version of MCN that inherits the network architecture from the CCN with $N = 16$ (number of feature blocks) and $S = 5$ (filter support). PixelCNN++ [22] is originally designed for image inpainting as a generative model. Here we adapt it for entropy modeling. Starting from a down-sampled grayscale image with a factor of four, we use a DNN of similar model complexity compared with our CCN to predict the categorical distributions from the down-sampled image. SINs summarize the side information of $\hat{y}$ with a separate DNN, which is helpful in probability estimation. We adopt a DNN-based autoencoder of similar model complexity as our CCN (including three stride convolutions and two residual connections) to generate the side information, which is further quantized and compressed with arithmetic coding for performance evaluation. We test five variants of SINs with different amount of side information by changing the number of output channels $M$. All competing models are trained on the same dataset described at the beginning of Section VI. We also introduce a fast version of our method, which we name CCNf, by making the network architecture lighter (with $N = 3$ and $S = 3$) and by dividing the input image into non-overlapping patches for parallel processing.

Fig. 7 shows the bit rates of the competing methods on the Kodak dataset. The proposed CCN matches the best performing model MCN, which suggests that with the proposed zigzag coding order and code dividing technique, CCN does not exclude the most important codes from the partial context of the
current code being processed. The bit rates of SINs come from two parts - the image itself and the side information. It is clear from the figure that increasing the amount of side information leads to bit savings of the image, at the cost of additional bits introduced to code the side information. In general, it is difficult to determine the amount of side information for optimal compression performance. PixelCNN++ can also be regarded as a special case of SINs, whose side information is a small image without further processing, leading to the worst performance.

We also compare CCN with the widely used lossless image compression standards, including TIFF, GIF, PNG, JPEG-LS, JPEG2000-LS, and BPG. All test results are generated by MATLAB2017. From Fig. 8, we find that CCN (along with its fast version CCN) overwhelms all competing methods on the Kodak/Tecnick dataset, achieving more than 5 times speed up compared to the best lossless image compression standard, JPEG2000-LS.

The running time of the four types of DNN-based entropy models is tested on an NVIDIA TITAN Xp machine, whose results on the Kodak dataset are listed in Table I. For encoding, CCN enjoys the fastest speed, followed by PixelCNN++ and SIN (the best performing variant). Despite similar encoding time, they have substantially different decoding complexities. PixelCNN++ has the fastest decoder, followed by SIN. Due to the sequential decoding nature, MCN is the slowest, taking nearly one hour to decode a grayscale image of size $752 \times 496$. Our CCN achieves a significant improvement upon MCN with the proposed code dividing technique, while maintaining nearly the same bit rates. Moreover, CCN speeds up CCN more than 30 times, striking a good balance model between efficiency and accuracy.

We conduct thorough ablation experiments to analyze the impact of individual components to final compression performance. Fig. 9 shows the bit rates of CCNs with three different numbers of feature blocks ($N \in \{4, 8, 16\}$) and two filter sizes ($S \in \{3, 5\}$). When the filter size and the network depth are fixed, adding more feature blocks effectively increases the model capability and thus boosts the compression performance. Similarly, using a larger filter size with fixed network depth and feature block number increases the partial context, leading to better entropy modeling. Moreover, we replace the proposed zigzag coding order in CCN with the raster coding order, whose model is denoted by CCN. From Fig. 9, we observe that the performance of CCN drops significantly, only comparable to the CCN with four feature blocks, which verifies the advantages of the proposed coding order.

B. Lossy Image Compression

In lossy image compression, the analysis transform $g_a$, the non-uniform quantizer $g_d$, the CCN-based entropy model, and the synthesis transform $g_s$ are jointly optimized for...
Fig. 13. Compressed images by different compression methods on the Kodak dataset. The quantitative measures are in the format of “bpp / PSNR / MS-SSIM”. (a) Uncompressed “Sailboat” image. (b) Ballé17 [8]. 0.209 / 31.81 / 0.962. (c) Li18 [23]. 0.244 / 31.97 / 0.966. (d) BPG. 0.220 / 33.19 / 0.963. (e) Ours optimized for MS-SSIM. 0.209 / 31.01 / 0.978. (f) Uncompressed “Statue” image. (g) Ballé17. 0.143 / 29.48 / 0.942. (h) Li18. 0.115 / 29.35 / 0.938. (i) BPG. 0.119 / 29.77 / 0.935. (j) Ours optimized for MS-SSIM. 0.116 / 28.05 / 0.954.

Fig. 14. Compressed images by different compression methods on the Tecnick dataset. The quantitative measures are in the format of “bpp / PSNR / MS-SSIM”. (a) Uncompressed “Bus” image. (b) JPEG2K. 0.199 / 24.41 / 0.914. (c) Li18 [23]. 0.224 / 23.41 / 0.908. (d) BPG. 0.208 / 25.36 / 0.928. (e) Ours(MS-SSIM). 0.198 / 23.71 / 0.951. (f) Uncompressed “Toy train” image. (g) JPEG2K. 0.201 / 28.29 / 0.917. (h) Li18. 0.189 / 26.83 / 0.899. (i) BPG. 0.210 / 29.25 / 0.933. (j) Ours(MS-SSIM). 0.198 / 28.08 / 0.949.
with a learning rate of $\lambda = 8$, and the number of Gaussian components in MoG is $C = 3$. Fig. 10 shows the learned continuous distributions of some codes, which are typically complex and multimodal. This optimization is performed separately for each $\lambda$ and for each distortion measure. We optimize twelve models for six bit rates and two distortion metrics (MSE and MS-SSIM). MSE is converted to peak signal-to-noise ratio (PSNR) for quantitative analysis.

We compare our methods with existing image coding standards and recent DNN-based compression models. These include JPEG [14], JPEG2000 [15], BPG [42], Agustsson17 [17], Theis17 [16], Toderici17 [28], Rippel17 [18], Mentzer18 [25], Johnston17 [29], Balle17 [8], and Li18 [23]. Both JPEG (with 4:2:0 chroma subsampling) and JPEG2000 are based on the optimized implementations in MATLAB2017. For BPG, we adopt the latest version from its official website. For the lossy case, our method offers improvements both in the lossless case, our method achieves the best compression rates and two distortion metrics (MSE and MS-SSIM). MSE is not available. Therefore, we report the results from their respective papers.

Fig. 11 shows the rate-distortion curves on the Kodak dataset. We find that our method optimized for MSE outperforms all competing methods at low bit rates (< 0.5 bpp), except for BPG. When optimized for MS-SSIM, our method performs on par with Rippel17 and is much better than the rest. Fig. 12 shows the rate-distortion curves on the Tecnick dataset, where we observe similar trends for both PSNR and MS-SSIM. An interesting observation is that when we continue increasing the bit rate, PSNR/MS-SSIM starts to plateau, which may be due to the limited model capability. Without any constraint on rate ($\lambda = \infty$) and quantization ($L = \infty$), our method optimized for MSE only reaches 38.2dB on the Kodak dataset, which may be considered as an empirical upper-bound for our network structure. Preliminary results indicate that increasing the depth and width of the network leads to performance improvements at high bit rates.

We visually compare the compressed images by our method against Balle17, Li18, JPEG2K, and BPG. Fig. 13 and Fig. 14 show sample compressed results on the Kodak and Tecnick datasets, respectively. JPEG2K and BPG exhibit artifacts (such as blocking, ringing, blurring, and aliasing) that are common to all linear transform coding methods, reflecting the underlying linear basis functions. Ballé17 is effective at suppressing ringing artifacts at the cost of over-smoothing fine structures. Li18 allocates more bits to preserve large-scale strong edges, while tends to eliminate small-scale localized features (e.g., edges, contours, and textures). In contrast, our method generates compressed images with more faithful details and less visible distortions. In addition, we investigate the effectiveness of joint optimization of the transforms and the CCN-based entropy model. A baseline method is introduced, which first optimizes the transforms for MSE ($\lambda = \infty$), and trains the CCN-based entropy model with learned (and fixed) transforms. As shown in Fig. 11 (a), the baseline method underperforms the jointly optimized one by 1dB.

We report the running time of our method at six bit rates on the Kodak dataset using the same machine. It takes 0.024 second to generate $\hat{y}$ and 0.032 second to reconstruct the image. The entropy coding time is listed in Table II, which we see that more time is needed to encode and decode images at higher bit rates. With the help of the proposed code dividing technique, our method performs entropy decoding in around one second for images of size $752 \times 496$.

### VII. Conclusion and Discussion

We have introduced CCNs for context-based entropy modeling. Parallel entropy encoding and decoding are achieved with the proposed coding order and code dividing technique, which can be efficiently implemented using mask convolutions. We test the CCN-based entropy model (combined with arithmetic coding) in both lossless and lossy image compression. For the lossless case, our method achieves the best compression performance, which we believes arises from the more accurate estimation of the Bernoulli distributions of the binary codes. For the lossy case, our method offers improvements both visually and in terms of rate-distortion performance over image compression standards and recent DNN-based models.

The application scope of the proposed CCN is far beyond building the entropy model in image compression. As a general probability model, CCN appears promising for a number of image processing applications. For example, we may use CCN to learn a probability model $P(x)$ for natural images, and use it as a prior in Bayesian inference to solve various vision tasks such as image restoration [43], [44], image quality assessment [45], [46], and image generation [10], [47].

### Acknowledgment

The authors would like to thank the NVIDIA Corporation for donating a TITAN Xp GPU used in this research.

### REFERENCES

[1] Wikipedia, “Morse code,” 2019. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Morse_code&oldid=895323877

[2] D. A. Huffman, “A method for the construction of minimum-redundancy codes,” *IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[3] I. H. Witten, R. M. Neal, and J. G. Cleary, “Arithmetic coding for data compression,” *Commun. ACM*, vol. 30, no. 6, pp. 520–540, 1987.
[4] G. Martin, “Range encoding: An algorithm for removing redundancy from a digitised message,” in Video & Data Recording Conf., 1979, pp. 24–27.

[5] C. E. Shannon, “A mathematical theory of communication,” Bell System Tech. J., vol. 27, no. 3, pp. 379–423, 1948.

[6] N. Ahmed, T. Natarajan, and K. R. Rao, “Discrete cosine transform,” IEEE Trans. Comput., vol. C-23, no. 1, pp. 90–93, 1974.

[7] R. A. DeVore, B. Jawerth, and B. J. Lucier, “Image compression through wavelet transform coding,” IEEE Trans. Inform. Theory, vol. 38, no. 2, pp. 719–746, 1992.

[8] J. Ballé, V. Larapara, and E. P. Simoncelli, “End-to-end optimized image compression,” in Int. Conf. Learning Representations, 2017.

[9] D. Marpe, H. Schwarz, and T. Wiegand, “Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard,” IEEE Trans. Circuits Syst. Video Technol., vol. 13, no. 7, pp. 620–636, 2003.

[10] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel recurrent neural networks,” in Int. Conf. Machine Learning, 2016, pp. 1747–1756.

[11] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, “Conditional image generation with PixelCNN decoders,” in Neural Inf. Process. Syst. 2016, pp. 4797–4805.

[12] T. M. Cover and J. A. Thomas, Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing). New York, NY, USA: Wiley-Interscience, 2006.

[13] R. Suddhar, K. Rathaiga, and S. Jayaraman, “Image compression using coding of wavelet coefficients—a survey,” ICGST-GVIP J., vol. 5, no. 6, pp. 25–38, 2005.

[14] G. K. Wallace, “The JPEG still picture compression standard,” IEEE Trans. Consumer Electron., vol. 38, no. 1, pp. 18–34, 1992.

[15] A. Skodras, C. Christopoulos, and T. Ebrahimi, “The JPEG 2000 still image compression standard,” IEEE Signal Process. Mag., vol. 18, no. 5, pp. 36–58, 2001.

[16] L. Theis, W. Shi, A. Cunningham, and F. Huszár, “Lossy image compression with compressive autoencoders,” in Int. Conf. Learning Representations, 2017.

[17] E. Agustsson, F. Mentzer, M. Tschannen, L. Cavigelli, R. Timofte, L. Theis, W. Shi, A. Cunningham, and F. Huszár, “Lossy image compression with compressive autoencoders,” in IEEE Conf. Comput. Vis. Pattern Recog., 2016, pp. 10 794–10 803.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in Neural Inf. Process. Syst. 2014, pp. 2672–2680.

[19] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in IEEE Conf. Comput. Vis. Pattern Recog., 2016, pp. 770–778.

[20] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in IEEE Conf. Comput. Vis. Pattern Recog., vol. 1, no. 2, 2017, pp. 4700–4708.

[21] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, “Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network,” in IEEE Conf. Comput. Vis. Pattern Recog., 2016.

[22] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binary neural networks: Training deep neural networks with weights and activations constrained to +1 or -1,” arXiv:1602.02830, 2016.

[23] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, “DoReFa-Net: Training low bitwidth convolutional neural networks with low bitwidth gradients,” arXiv:1606.06160, 2016.

[24] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “Xnor-net: Imagenet classification using binary convolutional neural networks,” in Eur. Conf. Comput. VIs., 2016, pp. 525–542.

[25] S. Lloyd, “Least squares quantization in PCM,” IEEE Trans. Inform. Theory., vol. 28, no. 2, pp. 129–137, 1982.

[26] N. Asuni and A. Giachetti, “TESTIMAGES: A large data archive for display and algorithm testing,” Journal of Graphics Tools, vol. 17, no. 4, pp. 113–125, 2013.

[27] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Int. Conf. Learning Representations, 2015.

[28] F. Bellard, “BPG image format,” 2019. [Online]. Available: https://bellard.org/bpg/

[29] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, “Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising,” IEEE Trans. Image Process., vol. 26, no. 7, pp. 3142–3155, 2017.

[30] J. Portilla, V. Strela, M. J. Wainwright, and E. P. Simoncelli, “Image denoising using scale mixtures of Gaussians in the wavelet domain,” IEEE Trans. Image Process., vol. 12, no. 11, 2003.

[31] Z. Wang and A. Bovik, Modern Image Quality Assessment, 1st ed. Morgan & Claypool Publishers, 2006.

[32] K. Ma, W. Liu, K. Zhang, Z. Duanmu, Z. Wang, and W. Zuo, “End-to-end blind image quality assessment using deep neural networks,” IEEE Trans. Image Process., vol. 27, no. 3, pp. 1202–1213, 2018.

[33] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, “Neural ordinary differential equations,” in Neural Inf. Process. Syst. 2018, pp. 6571–6583.