

**Block-based Image Coding with Autoencoder and
Border Information**

THESIS

Submitted in Partial Fulfillment of

the Requirements for

the Degree of

MASTER OF SCIENCE (Electrical Engineering)

at the

NEW YORK UNIVERSITY

TANDON SCHOOL OF ENGINEERING

by

Zhongzheng Yuan

May 2020

**Block-based Image Coding with Autoencoder
and Border Information**

THESIS

**Submitted in Partial Fulfillment of
the Requirements for
the Degree of**

MASTER OF SCIENCE (Electrical Engineering)

at the

**NEW YORK UNIVERSITY
TANDON SCHOOL OF ENGINEERING**

by

Zhongzheng Yuan

May 2020

Approved:



Advisor Signature

5/19/2020

Date



Department Chair Signature

May 20, 2020

Date

University ID: **N15965747**
Net ID: **zy740**

Approved by the Guidance Committee:

Major: Electrical Engineering



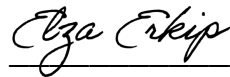
Yao Wang

Professor

Electrical and Computer Engineering

5/19/2020

Date



Elza Erkip

Professor

Electrical and Computer Engineering

May 19, 2020

Date



Yong Liu

Professor

Electrical and Computer Engineering

May 19, 2020

Date

Vita

Zhongzheng (Jacky) Yuan was born in Guangzhou China. He started his undergraduate education in New York University Tandon School of Engineering in September 2015 and graduated with a Bachelor of Science degree in Electrical Engineering. He continued his graduate education in NYU starting in September 2019. He studied this thesis in NYU Video Lab from May 2019 to May 2020.

Acknowledgements

I would like to thank my advisor Prof. Yao Wang for all her support, valuable insights, and encouragement through the past year. Her guidance helped me in all the time of this research and writing of this thesis, and her rigorous attitude toward research has always inspired me to improve.

I am grateful to Prof. Elza Erkip and Prof. Yong Liu for serving on my committee and giving valuable advices.

I would like to thank our lab member Jeffrey Mao, who worked on this research project with me and having done some of the initial study on the denoiser model. His support and encouragements kept me moving forward at times when I am in frustration. I would also like to thank visiting member to our lab, Haojie Liu, for his experienced advices on deep learning and compression.

Last, but not least, I would like to thank my parents for their unconditional love and care, and always being there for me when I need the warmth of home.

ABSTRACT

Block-based Image Coding with Autoencoder and Border Information

by

Zhongzheng Yuan

Advisor: Prof. Yao Wang

Submitted in Partial Fulfillment of the Requirements for
the Degree of Master of Science (Electrical Engineering)

May 2020

In this work, we build upon the image compression framework developed by Ballé et al. and adapt it for block-based image coding. An autoencoder is trained to learn forward and inverse transforms to encode image blocks into latent features. When the autoencoder model is directly applied on small blocks, spatial redundancy cannot be fully utilized. Therefore, it is crucial to use reconstructed border pixels to exploit spatial redundancy across adjacent blocks. We propose three different models to incorporate border information to the autoencoder framework: the first using a post-decoder denoiser, the second using border pixels to predict latent features, and the third using border pixels to predict parameters for the distribution of latent features. We discussed the procedure to train these models and evaluated their rate-distortion performance. Our final model has a higher rate-distortion performance than JPEG2000 but is slightly below Ballé's model.

Table of Contents

Vita.....	iii
Acknowledgements.....	iv
Abstract.....	v
Table of Contents.....	vi
List of Figures.....	vii
1. Introduction.....	1
1.1 Background.....	1
1.2 Related Works.....	1
1.3 Block-based coding.....	2
1.4 Training and Evaluation Methodology.....	3
1.5 Outline.....	4
2. Autoencoder Adapted for Block-based Coding.....	6
3. Autoencoder with Denoiser.....	8
4. Difference Network.....	11
5. Entropy Parameters Prediction Network.....	15
6. Compression Performance Evaluation.....	18
6.1 Using reconstructed border in decoding.....	18
6.2 Training using noisy/decoded border.....	20
7. Conclusion.....	23
7.1 Summary of results.....	23
7.2 Future work.....	23
Appendix.....	25
Bibliography.....	26

List of Figures

Figure 1.1: Layout of a 32x48 patch.....	4
Figure 2.2: Autoencoder structure with 16x16 block input	7
Figure 2.2: Autoencoder performance on 16x16 blocks.....	7
Figure 3.1: Autoencoder + Denoiser model.....	8
Figure 3.2: Performance of Autoencoder + Denoiser.....	9
Figure 3.3: Sample outputs from Autoencoder + Denoiser	10
Figure 4.1: Border prediction module.....	12
Figure 4.2: Difference Network.....	12
Figure 4.3: Difference Network performance.....	13
Figure 4.4: Sample outputs of Difference Network.....	14
Figure 5.1: Entropy Parameter Prediction Network	16
Figure 5.2: Performance of all three networks.....	17
Figure 6.1: Degredation in performance when border is decoded pixels	18
Figure 6.2: Sample decoded image 1 using network trained with clean border	18
Figure 6.3: Sample decoded image 2 using network trained with clean border	20
Figure 6.4: Comparison of performance.....	20
Figure 6.5: Sample decoded image 1 using network trained with decoded border	20
Figure 6.6: Sample decoded image 2 using network trained with decoded border	21

1. Introduction

1.1 Background

Image compression is an important and widely studied problem in engineering. With the increasing prevalence of high-definition image and video contents, there is a need to improve image compression efficiency beyond current image coding standards. Image compression is usually achieved by transforming the input image into a more energy compact latent space. Then the latent space data can be quantized and losslessly compressed into shorter bit-stream. The coded bit-stream is decoded and inverse transformed back to image domain on the decoder side. The goal of an image coder is to minimize the error caused by coding, while also minimizing the bits needed to code the image.

Deep learning-based coding methods have shown to have higher performance than image coding standards such as JPEG2000 which uses discrete wavelet transformation. Deep learning-based coding allows for a set of nonlinear transformation to be learned for more efficient mapping of pixels from image into a latent space than linear transforms used by traditional image codecs. This can be done with an autoencoder structure, where encoder and decoder are trained to learn the forward and inverse transforms.

1.2 Related Works

A deep learning based autoencoder compression framework was developed by Ballé et al., where the encoder and decoder were constructed using convolutional layers with GDN activation [1]. An end-to-end optimization of the network's rate-distortion

performance was achieved by minimizing both rate and distortion loss through gradient descent with the loss function:

$$R + \lambda \cdot D = E_{x \sim p(x)}[-\log_2 p_{\hat{y}}(\hat{y})] + \lambda \cdot E_{x \sim p(x)}\|x - \hat{x}\| \quad (1),$$

where \hat{y} is the quantized latent tensor, and x and \hat{x} are original and decoded image, and λ is the Lagrange multiplier that determines rate-distortion trade-off. The latent tensor y is modeled as a gaussian random variable, and the quantized latent \hat{y} is losslessly compressed with entropy coder. To allow for gradient propagation, quantization was simulated by adding a uniform gaussian noise to the latent tensors during training.

Another work by Ballé et al. extend upon the above framework by adding a hyperprior network for more efficient entropy coding [2]. The hyperprior network encodes a hyper-latent tensor z as side information. The hyper-latent provides a hyperprior for the gaussian probability model used in entropy coding of the latent tensor \hat{y} .

1.3 Block-based coding

In the deep learning framework developed by Ballé et. al., as well subsequent studies based on that framework, a whole image was taken as input to the autoencoder [3]. While coding the whole image is convenient and effective for a single image, it is also possible to code an image by dividing the image into small blocks and individually code each block. Block-based coding makes it possible to incorporate inter-frame temporal prediction for video coding, and is widely adopted in international video coding standards.

In this work, we will adapt the autoencoder with hyperprior structure to perform block-based image coding and utilize border information to improve coding performance. We will consider the Border Pixels region shown in Figure 1.1 as the border for the 16x16 target block. In the coding of each 16x16 block in an image, our models will take a 32x48 patch that encloses the target block as input, and information is extracted from the border to help code the target block.

1.4 Training and Evaluation Methodology

The standard test image set Kodak was used as a test set [8]. To evaluate the performance of our models developed in the subsequent chapters, we use Peak Signal to Noise Ratio (PSNR)

$$PSNR = 10 \log_{10} \frac{255^2}{MSE}, \quad MSE = ||x - \hat{x}||_2^2 \quad (2)$$

and bits per pixel (BPP) estimated by entropy

$$BPP = E_{x \sim p(x)}[-\log_2 p_{\hat{y}}(\hat{y})] + E_{x \sim p(x)}[-\log_2 p_{\hat{z}}(\hat{z})] \quad (3)$$

as evaluation criteria.

These metrics are calculated for each decoded image in the Kodak dataset and the average PSNR and BPP among all images is taken to be the performance of a model. One important note is that although BPP is calculated through entropy estimate, we have experimentally verified that the practical bit rate: length of coded bit stream divided by number of pixels, is very close to the estimated entropy (within 2%) when a range coder is employed to perform entropy coding.

In our initial testing of the models, we made the assumption that border pixels are perfectly reconstructed, which means during decoding, border pixel values are taken from the original image. This assumption was made for the ease of evaluation and comparison, but it leads to higher performance than what is actually achievable. In chapter 6, we will discuss this in more detail.

To train the models, we use the COCO dataset which has an abundance of images with similar resolution as Kodak test images [7]. Blocks of 32x48 were randomly extracted from COCO as training examples. The final train set has 200,000 blocks. Adam optimizer was used with a learning rate of 10^{-4} . The batch size was set to 50. Each network was trained for 200 epochs and convergence was generally reached within 200 epochs.

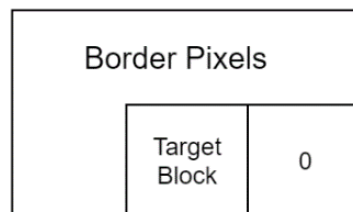


Figure 1.1: Layout of a 32x48 patch. The middle 16x16 block is the target. 16 rows and columns to the top and left of the target block is considered as border. The unknown pixels to the right are filled with zero.

1.5 Outline

The thesis is organized as follows:

In chapter 2, we will adapt the autoencoder structure directly on 16x16 blocks without using border information and compare the performance to whole-image coding.

In chapter 3-5, we propose three different models to incorporate border information to the autoencoder structure and evaluate their performance.

In chapter 6, we conduct a more accurate evaluation our final model by using decoded border instead of clean border when decoding an image.

In chapter 7, we conclude the findings in this thesis and discuss future directions of research.

2. Autoencoder Adapted for Block-based Coding

Based on the autoencoder structure in [2], we made some modifications to adapt it for block-based coding. The number of strided convolution layers was decreased to two for both the encoder and decoder to account for the small block size. In our initial experiments, we have tried to replace GDN by leaky ReLU activation followed by batch normalization. This led to a 0.5dB decrease in PSNR, in line with the result in [5] where GDN activation was found to have a better performance. We have also tried to decrease the number of channels to 64 and 32 for a more light-weight network. However, it was found that to achieve comparable results, the channel size has to remain large.

Encoder	Decoder	Hyper Encoder	Hyper Decoder
Conv: 3x3 c192 s1 GDN	Deconv: 3x3 c192 s2 IGDN	Conv: 3x3 c192 s1 Leaky ReLU	Deconv: 3x3 c192 s2 Leaky ReLU
Conv: 3x3 c192 s2 GDN	Deconv: 3x3 c192 s1 IGDN	Conv: 3x3 c192 s2 Leaky ReLU	Deconv: 3x3 c192 s2 Leaky ReLU
Conv: 3x3 c192 s1 GDN	Deconv: 3x3 c192 s2 IGDN	Conv: 3x3 c192 s2	Deconv: 3x3 c192 s1
Conv: 3x3 c192 s2	Deconv: 3x3 c192 s1		

Table 1. Layer details for autoencoder. Each row corresponds to a layer in the respective network module. For example, 3x3 c192 s1 means that a 3x3 kernel is used, number of channels is 192, and the stride is 1. GDN stands for generalized divisive normalization, and IGDN stands for inverse GDN.

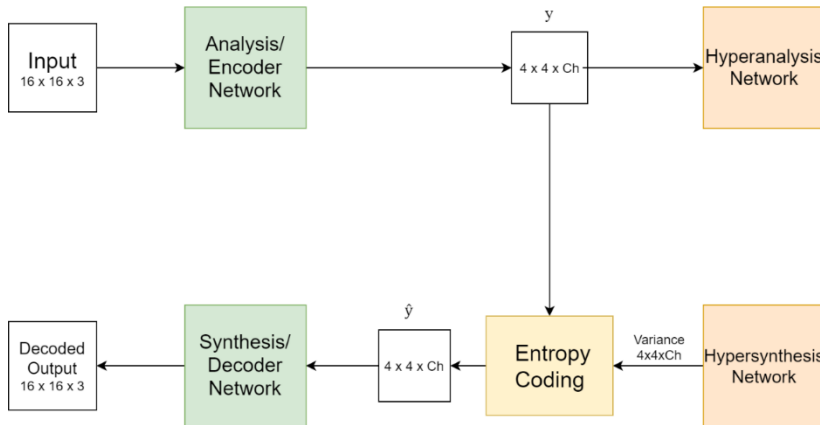


Figure 2.1: Autoencoder structure with 16x16 block input

With this setup, we trained the model for four different lambda values, each correspond to a point on the RD curve. The resulting performance is shown in Figure 2.2. The performance is much lower than what was achieved by the Ballé model. This was expected since decreasing the input size to 16x16 prohibited the network to fully utilize spatial redundancy in an image. To achieve better performance, border information should be utilized.

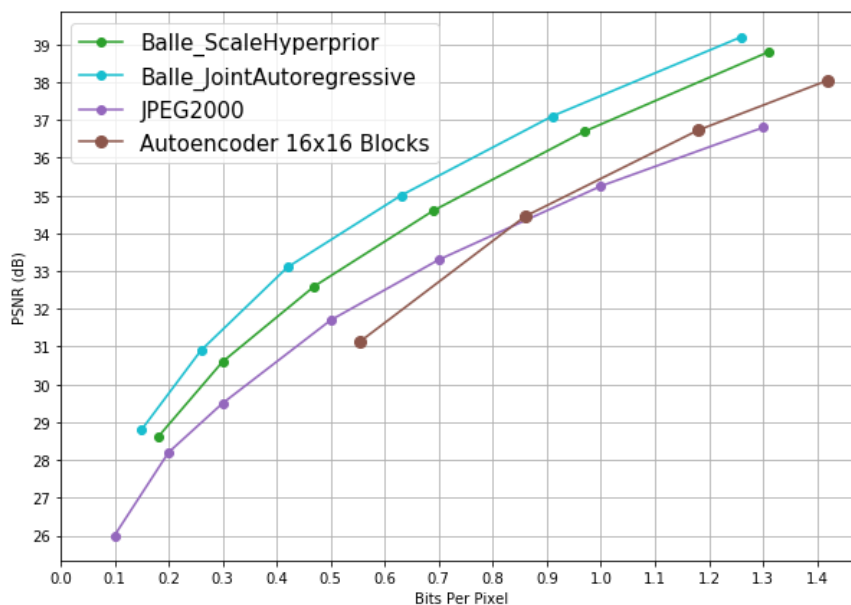


Figure 2.2 : Autoencoder performance on 16x16 blocks

3. Autoencoder with Denoiser

Our first attempt to incorporate border information into the network is to add a Denoiser network following the output of the autoencoder. The Denoiser network takes the decoded block and its border as inputs and perform noise removal on the decoded block while also trying to smooth the discontinuity between the block and its border. For the Denoiser network, we used the Group Residual Dense Network, which was shown to be effective in compression artifact reduction [5]. We reduced the number of RDBs to two to reduce complexity of the model.

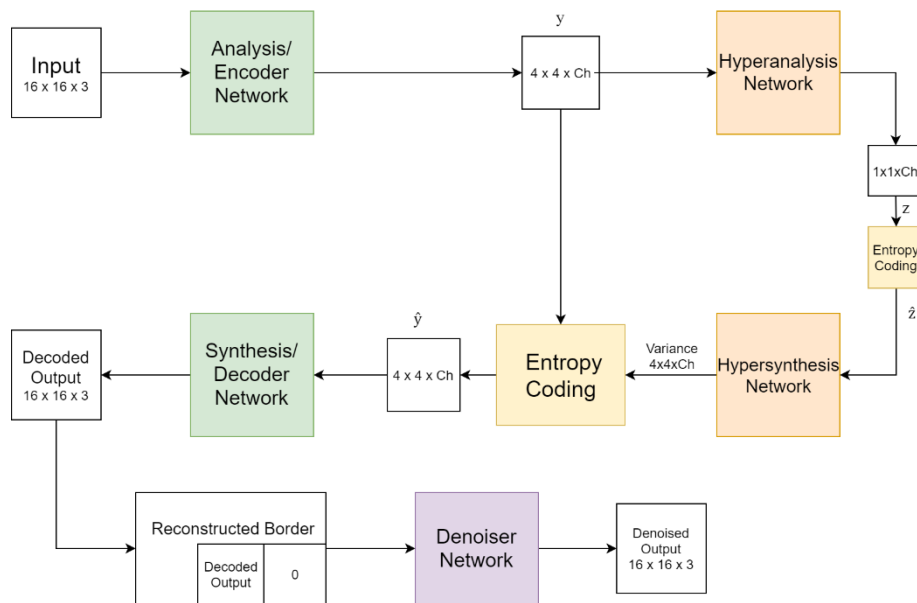


Figure 3.1: Autoencoder + Denoiser model

With the addition of Denoiser, the network learns to send less information that can be predicted from the border, leaving the Denoiser to recover those information when reconstructed border becomes available. The performance of this network is shown in Figure 3.2. Compared to the autoencoder alone, the denoiser achieved a higher PSNR and lower BPP. This improvement is very significant for low bit-rate models.

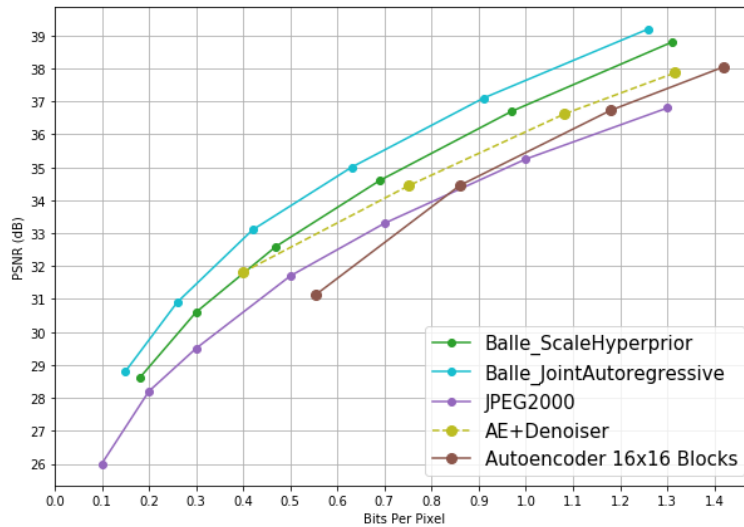


Figure 3.2: Performance of Autoencoder + Denoiser Network

Some sample outputs of this network are shown in Figure 3.3, the first column shows the original block before coding, the second column shows the pre-denoised block along with its border, and the third column shows the final denoised block. From the intermediate outputs, we can see that most of the details in the original block is present, even those edges that can be predicted from the border is sent with the latent tensor. Information such as color can be easily predicted from the border, so it is less accurately reconstructed from the latent tensor alone. This shows that the Autoencoder + Denoiser network is working as intended, but still not effective enough to reduce the amount of border information in the latent tensor.

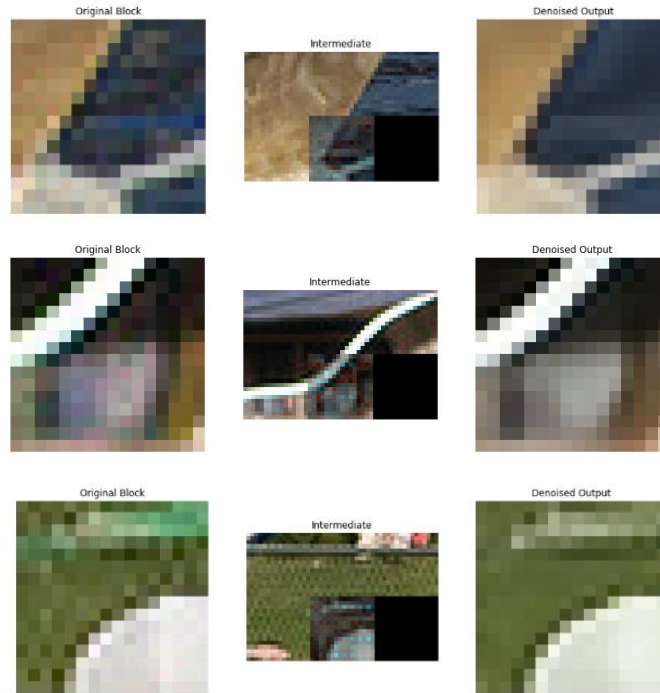


Figure 3.3: Sample outputs from Autoencoder + Denoiser Network

4. Difference Network

To help the autoencoder learn to encode less information from the border, we designed another network in which a residual of the latent tensor is encoded rather than the original latent. We first build a Border Prediction module to capture the information from border which can be used to predict the latent features from the autoencoder. This is achieved using six convolutional layers which takes the 32x48 border as input and output a 24-channel feature map. This process can be viewed as intra-prediction operation to fill in missing values for the zero region. And a large kernel size of 5x5 was chosen so that the receptive field is large enough to fill in those zero values. After intra-prediction, the 16x16 region in the feature map corresponding to the target block is taken out and encoded to 4x4x192 tensor by two strided convolution layers. The resulting tensor has the same size as the latent tensor.

We call this border predicted tensor y_{border} . And if the network is trained properly, it should predict the latent tensor as close as possible. To enforce this, the difference between the latent tensor y_{center} and y_{border} is taken, and the resulting difference tensor y_{diff} is entropy coded and sent to the decoder.

$$y_{diff} = y_{center} - y_{border} \quad (4)$$

By minimizing the bit-rate needed to code y_{diff} , the Border Prediction module learns to output a tensor that has a high correlation to y_{center} , so that their difference will have a lower variance and lower entropy.

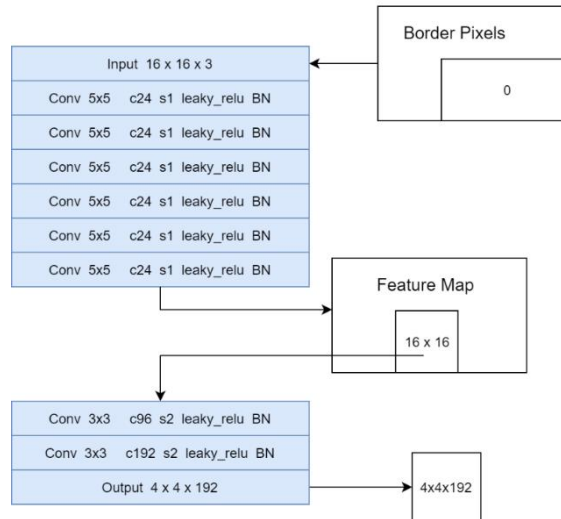


Figure 4.1: Border Prediction module, activation is leaky ReLU followed by batch normalization

The same operation to calculate y_{border} can be done at the decoder side using reconstructed border as input, and this reconstructed y_{border} is added to the decoded \hat{y}_{diff} to obtain \hat{y}_{center} .

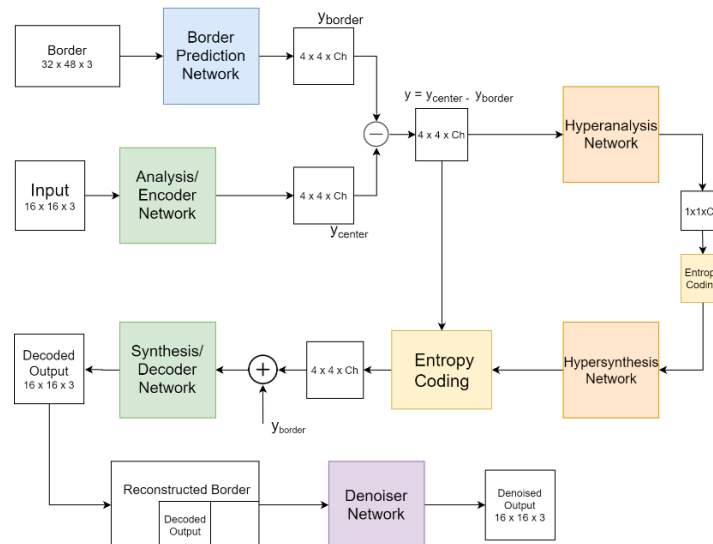


Figure 4.2: Difference Network

The performance of this network is shown in Figure 4.3. Compared to Autoencoder + Denoiser Network, PSNR increased by 0.27dB and BPP reduced by 0.071 on average.

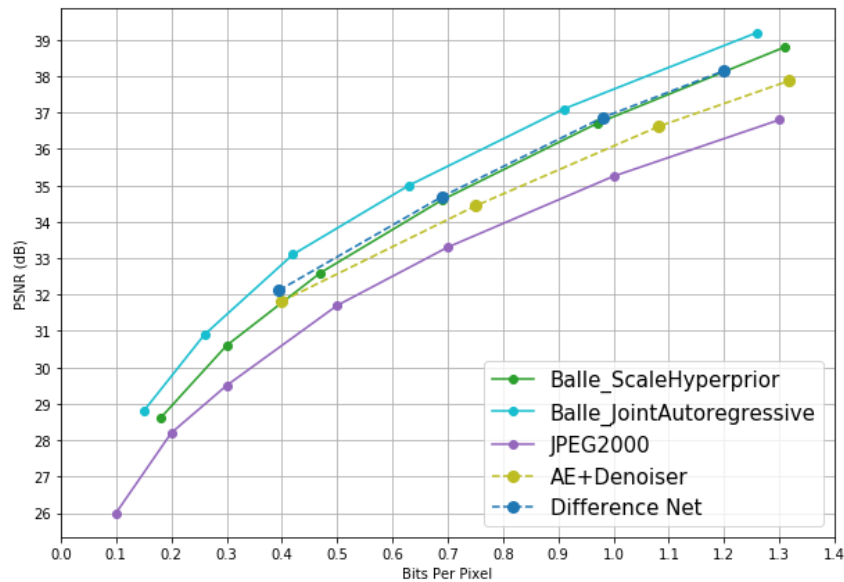


Figure 4.3: Difference Network performance

To demonstrate that the Border Prediction module is working properly to perform intra-prediction, we included some sample outputs in Figure 4.4. The third column shows the pre-denoised output block when only y_{border} is supplied as input to the decoder. In other words, no bit is used to send any information from the target block, and the output shows what can be predicted using border pixels alone. In the first and second examples, the border has clear edges that continue from the border into the target block. And using only the border pixels, the network was able to predict these edges accurately. In the third example, there is a red object in the center block that cannot be predicted from the border. The network predicts the most likely scenario that a straight edge continues from the

border. In this case, the prediction does not match the original block accurately, therefore, more bits would be needed to encode the presence of the red object.

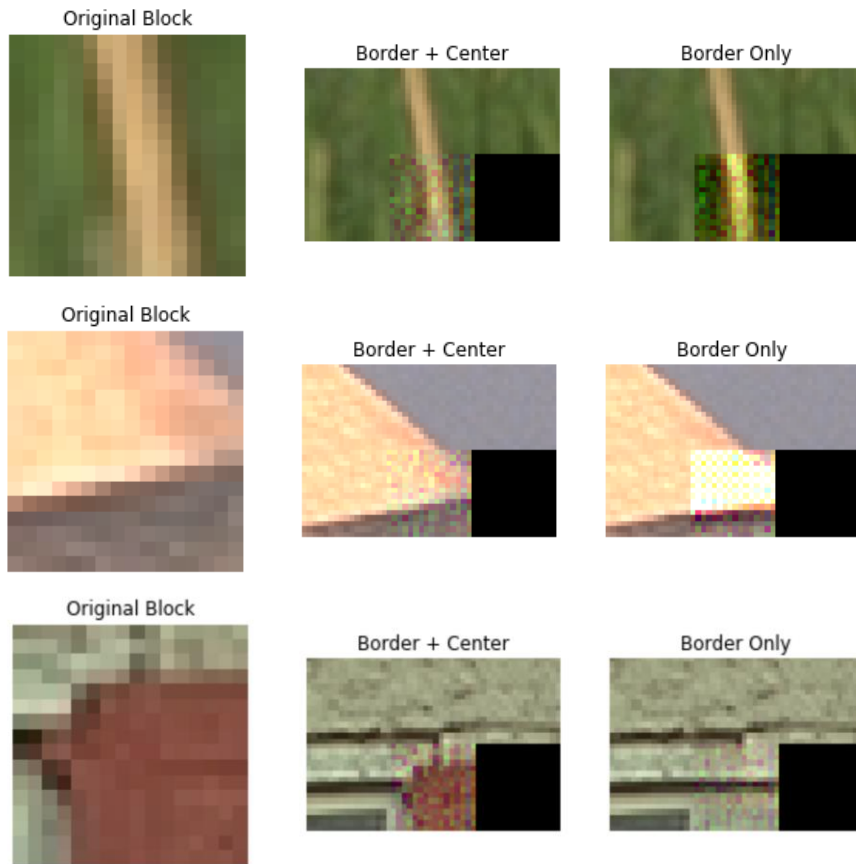


Figure 4.4: Sample outputs of the Difference Network. Original block (left column), pre-denoised outputs using $\hat{y}_{diff} + y_{border}$ to decode (middle column) and using only y_{border} to decode (right column)

5. Entropy Parameters Prediction Network

With the idea of generating prediction from border pixels, we implemented a third network that utilizes border to more accurately generate parameters for the probability model used in entropy coding. In the previous two networks, we have used only variance as parameter (and kept mean equal to zero) for the gaussian probability model for each element in the latent tensor. A more recent work has shown that modeling the gaussian probability using both mean and variance can further improve the RD performance [3].

In this network, we combine the hyper latent tensor and prediction from border pixels to generate mean and variance. This approach is an extension to the Difference Network. Because using border pixels to generate the mean is equivalent to the goal of the Difference Network, where the Border Prediction module tries to predict the latent tensor. Since the mean of a Gaussian probability is also the most probable value, generating the mean is same as predicting the latent tensor itself.

In addition, the hyper latent tensor is also entropy coded using gaussian model conditioned on the border pixels, whereas before, it used a non-parametric piecewise linear model which requires training and is fixed after training is done.

To predict entropy parameters, we added two networks (Mean and Variance Prediction Network A and B) that take the output from Border Prediction and generate the required entropy parameters for latent and hyper latent. Network A concatenates the tensors y_{border} and \hat{z} along the channel dimension and pass it through two layers of 1x1 convolutions with leaky ReLU activation. Network B takes y_{border} as input and uses 3x3 convolution of stride 2 to further downsample the 4x4 tensor to 1x1. The output channel

size is two times the latent channel size. One half of the output channel is used as mean, and the other half is used as variance.

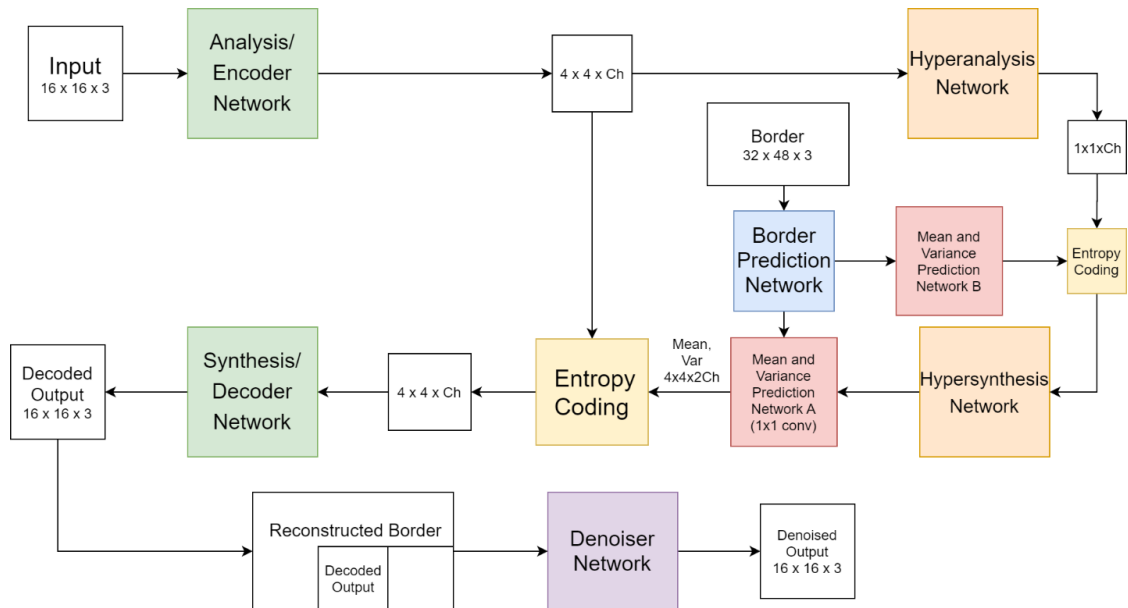


Figure 5.1: Entropy Parameter Prediction Network

In addition to this model, we have also tested a model where the hyperprior network was not used at all. Mean and variance parameters for y are directly predicted by the Border Prediction network. Although for this model, some bits were saved by not sending the hyper latent z , the mean and variance prediction from border alone are not as accurate as combining border and hyper latent. This model achieved the same performance as the Difference network, and therefore we chose the better performing model in Figure 5.1.

The performance of all three networks utilizing border information is shown in Figure 5.2. The performance of the Entropy Parameter Prediction Network is the highest among our three models.

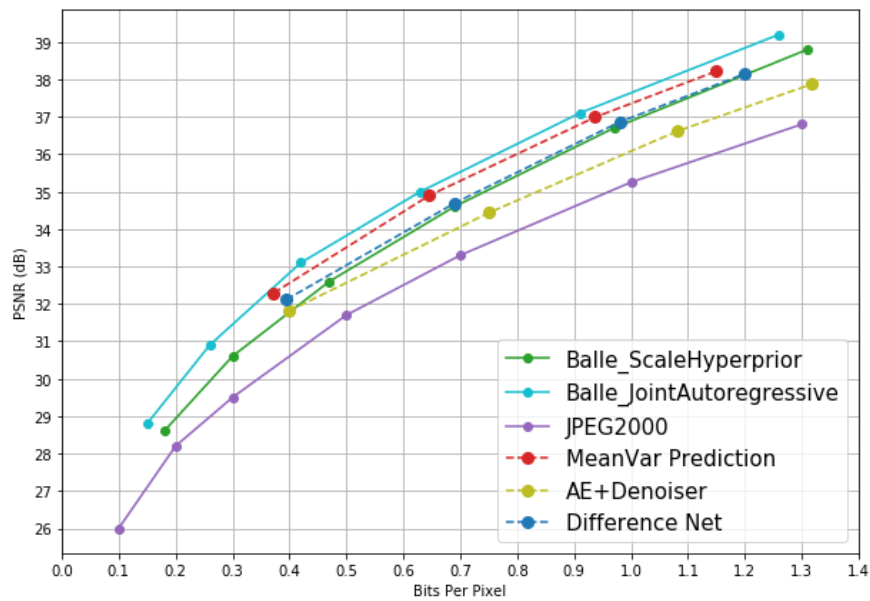


Figure 5.2: Performance of all three network utilizing border information

6. Compression Performance Evaluation

6.1 Using reconstructed border in decoding

In our previous experiments, we had made the assumption that border pixels are perfectly reconstructed for the ease of comparing different network structures. In this section, we conduct a more practical evaluation of network performance by decoding each block in an image using previous decoded blocks as border. Because of lossy compression, border pixels differ from their original once they are decoded. It was expected that this will incur some degradation in performance of the network.

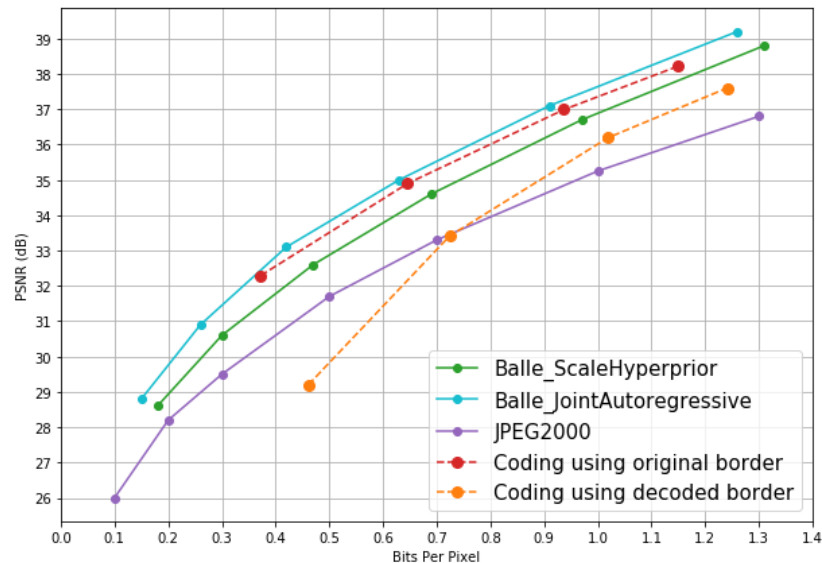


Figure 6.1: Degradation in performance when border is decoded pixels

As each block is decoded raster scan order, the difference of each block from their original accumulate, since in the decoding process of each block, the border pixels are used to denoise the current block. And when error in the border accumulate, the error in output blocks also increase correspondingly. Ideally, the autoencoder part of the network

should spend more bits to keep the error from growing, but because a denoiser is used at the end of the network, error from the border inevitably affect the target pixels.

We also observed that the performance degradation is more significant for the models trained for low bit rate than for higher bit rate models. This is because for higher rate models, the decoded pixels are closer to the original and the error propagated to succeeding blocks are less.

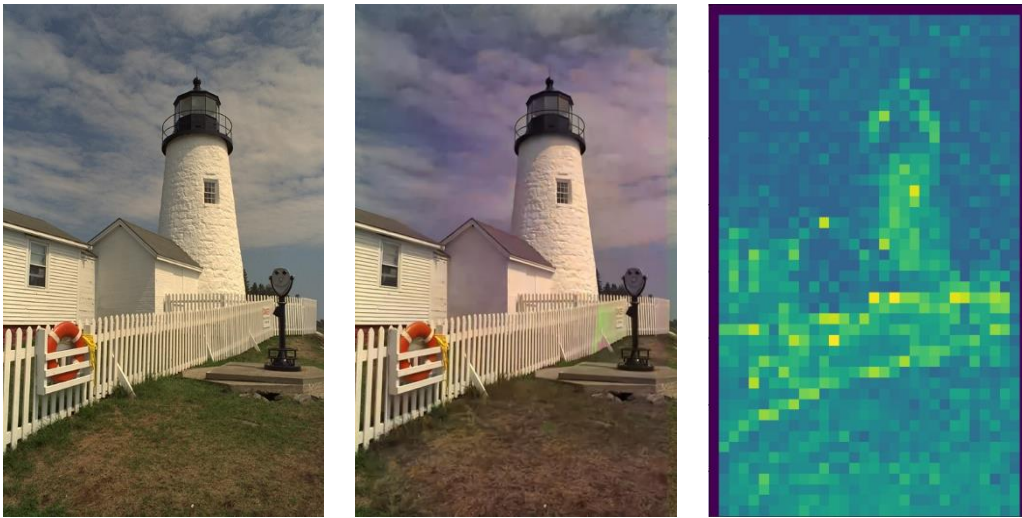


Figure 6.2: Sample decoded image 1 using network trained with clean border
Original Image (left), decoded Image (center), lengths of bit-stream for each block (right)
PSNR = 28.638 BPP = 0.439



Figure 6.3: Sample decoded image 1 using network trained with clean border
Original Image (left), decoded Image (center), lengths of bit-stream for each block
(right)

$$\text{PSNR} = 31.19 \quad \text{BPP} = 0.259$$

Two images from the Kodak dataset are decoded using the third network and are shown in Figures 6.2 and 6.3. The images have a color shift that is more severe towards the right side of the image, where error from decoding accumulates the most.

6.2 Training using noisy/decoded border

Although border error cannot be completely eliminated, we can train our network to suppress this error. In our previous training process, we have used error-free border as training examples. When the network encounter decoded input of a different distribution from the training set, it cannot generalize to these outliers. To make the network learn to account for coding error, we modified our dataset to purposefully include error in border pixels. We have only tested this with the $\lambda = 0.01$ model.

Because the coding error followed a gaussian distribution, one attempt was to add a gaussian noise to the border with the same variance, while the target block remained noise-free. This simple modification increased the PSNR by 0.73dB, and BPP reduced by 0.03.

Our second attempt was to use actual decoded pixels as border for training. First, we code images from the COCO dataset using a network trained on clean borders. Then we extract blocks from this processed dataset using the same procedure as before, but the border is replaced by the decoded version. This modified dataset is then used to train a new network. This approach achieved a higher gain in PSNR of 1.98dB, and BPP reduction by 0.06.

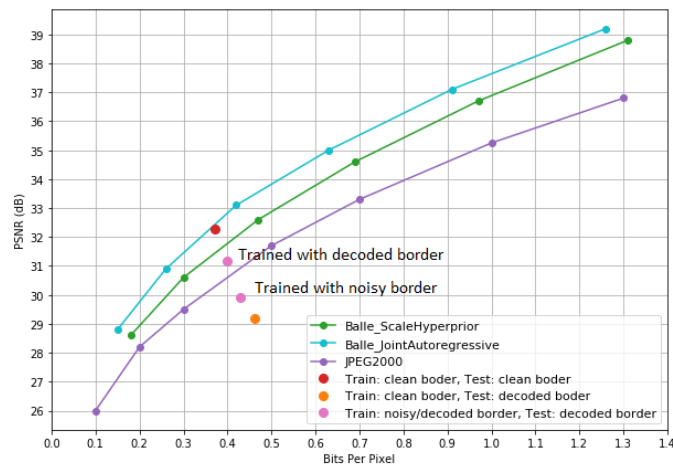


Figure 6.4: Comparison of performance when clean/decoded border is used in training and testing



Figure 6.5: Sample decoded image 1 using network trained with decoded border

Original Image (left), Decoded Image (right)

PSNR = 31.238 BPP = 0.360



Figure 6.6: Sample decoded image 2 using network trained with decoded border

Original Image (left), Decoded Image (right)

PSNR = 33.22 BPP = 0.196

In the sample decoded images above, color shift in the previous decoded samples are no longer present. Although PSNR for these two images are higher, block artifacts are significantly more visible in these images. Block artifacts are supposed to be removable by the denoiser, as it was designed to smooth the discontinuity between adjacent blocks. However, because of the way we modified our data, border pixels in the training samples are from a different distribution than the target block. When the denoiser train on these samples, it was unable to learn a set of filters that smooth this discontinuity. More investigation into how to address this block artifact is needed.

7. Conclusion

7.1 Summary of results

In this work, we adapted the autoencoder structure for whole-image coding to work on 16x16 block size. The main challenge is figuring out how to utilize border information to exploit spatial redundancy across adjacent blocks. We designed and evaluated three models to incorporate border information. Although our initial results show comparable performance to whole-image autoencoders, performance of our model degrades when error is present in reconstructed border. We partially mitigated this degradation by modifying the training dataset to account for this error.

7.2 Future work

The decoded images of our final model have a significant block artifact because of the modifications made to the training set. One possible improvement is to modify the training procedure to train parts of the model separately. First train a model without the denoiser and generate training data from this model. Then train the denoiser using such data, with a loss on the entire 32x48 patch to ensure that the output patch matches the original. This training procedure may help the denoiser to better smooth discontinuity between blocks.

The models trained in this work are all trained with fixed λ value and are optimized for a fixed bitrate. A recent study proposed a method to train a single model conditioned on a variable λ value [6]. And by tuning the λ value various bitrate can be achieved. Such

variable rate model will be more convenient for practical use than having to train and store separate models for different bitrate.

Finally, block-based structure allows us to easily extend this study to video coding. For example, we can add a component to our model for block-based temporal prediction. Features generated from temporal prediction can be combined with border prediction to predict probability distribution of latent features.

Appendix

Numerical values for model performance (trained and tested with clean border)

	$\lambda = 0.01$		$\lambda = 0.025$		$\lambda = 0.05$		$\lambda = 0.075$		Compare with Autoencoder Only (averaged over all λ)	
	BPP	PSNR	BPP	PSNR	BPP	PSNR	BPP	PSNR	BPP	PSNR
Autoencoder Only	0.554	31.13	0.860	34.45	1.18	36.73	1.42	38.05	-	-
Autoencoder + Denoiser	0.399	31.81	0.751	34.44	1.08	36.62	1.32	37.87	-0.116	+0.095
Difference Network	0.394	32.11	0.691	34.70	0.980	36.86	1.20	38.15	-0.187	+0.365
Entropy Parameters Prediction	0.372	32.29	0.646	34.90	0.937	36.99	1.15	38.22	-0.227	+0.51

Performance comparison when tested with decoded border

Entropy Parameters Prediction Network	$\lambda = 0.01$		$\lambda = 0.025$		$\lambda = 0.05$		$\lambda = 0.075$	
	BPP	PSNR	BPP	PSNR	BPP	PSNR	BPP	PSNR
Test with clean border	0.372	32.29	0.646	34.90	0.937	36.99	1.15	38.22
Test with decoded border	0.461	29.20	0.724	33.41	1.02	36.20	1.24	37.60

Bibliography

- [1] Ballé, Johannes, Valero Laparra, and Eero P. Simoncelli. "End-to-end optimized image compression." arXiv preprint arXiv:1611.01704 (2016).
- [2] Ballé, Johannes, et al. "Variational image compression with a scale hyperprior." arXiv preprint arXiv:1802.01436 (2018).
- [3] Minnen, David, Johannes Ballé, and George D. Toderici. "Joint autoregressive and hierarchical priors for learned image compression." *Advances in Neural Information Processing Systems*. 2018.
- [4] Ballé, Johannes. "Efficient nonlinear transforms for lossy image compression." 2018 *Picture Coding Symposium (PCS)*. IEEE, 2018.
- [5] Cho, Seunghyun, et al. "Low Bit-rate Image Compression based on Post-processing with Grouped Residual Dense Network." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019.
- [6] Choi, Yoojin, Mostafa El-Khamy, and Jungwon Lee. "Variable Rate Deep Image Compression With a Conditional Autoencoder." *Proceedings of the IEEE International Conference on Computer Vision*. 2019.
- [7] Lin, Tsung-Yi, et al. "Microsoft coco: Common objects in context." *European conference on computer vision*. Springer, Cham, 2014.
- [8] Kodak. <http://r0k.us/graphics/kodak/>. 1999.