Deep Neural Network Based Image Compression Encoders

Ruiqi Liu
rxl5448@psu.edu
Pennsylvania State University
USA

Yuxuan Liu
yxl429@psu.edu
Pennsylvania State University
USA

Srikanth Mantravadi
sxm6373@psu.edu
Pennsylvania State University
USA

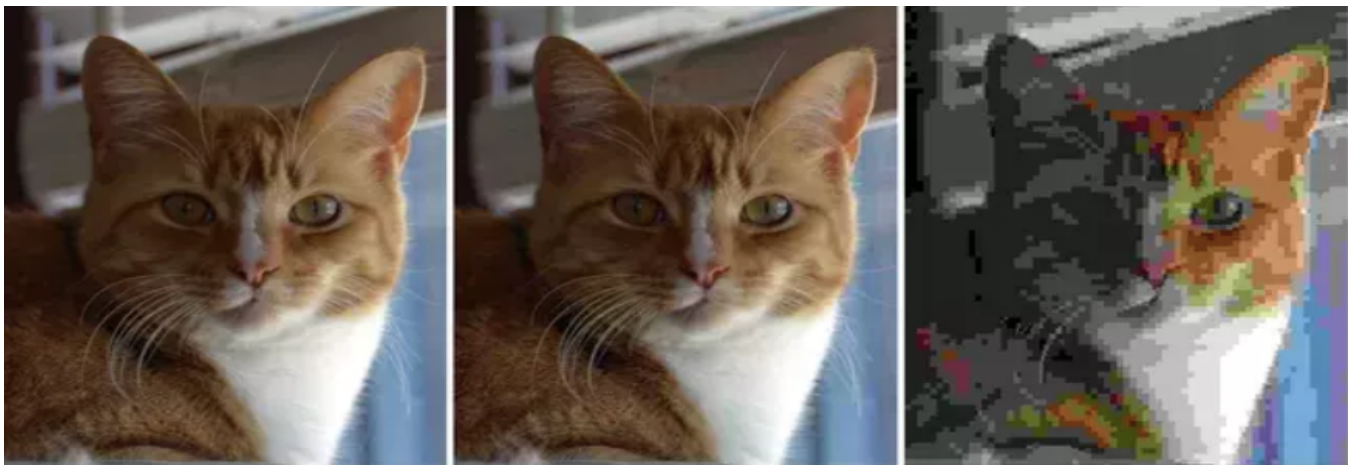

Figure 1. Three Levels of JPEG compression. The leftmost image is the original. The middle image offers a medium compression, which may not be immediately obvious to the naked eye without closer inspection. The rightmost image is maximally compressed.(https://lifewire.com/the-effect-of-compression-on-photographs-493726).

Abstract

Lossy image compression has become a rising topic in the community especially after deep learning is applied to it. As proved by Strumpler *et al* [13], building robust DNN based image compression encoders without needing modification on the decoder end i.e. encoders that work with traditional JPEG decoders is now a reality. We try to improve this work with our contributions: 1. Implement MS-SSIM in the original loss function and experiment on optimal parameters. 2. Replace VGG in the original network with ResNet. 3. Analyze inference time and influence of training steps. 4. Add LSTM into the attention network.

Keywords: Image Compression, Deep Neural Networks, Discrete Cosine Transform

1 Introduction

The interest in image compression has increased in the past few decades as there is huge amount of data being generated everywhere. The typical image resolution becomes larger and larger, so it becomes a must to find a reliable method to compress those images for both transmitting and storing purpose. The traditional method for image compression is use discrete cosine transform[18], this became the core algorithm for many future researches[13]. With the development of deep neural network these years, many researchers are trying to apply DNN to image compression tasks to improve rate-distortion performance.

Our paper focus on optimizing the image encoder without modifying the decoder side. We work on a baseline framework[13], modifying the loss function, changing the attention extraction mechanism and using LSTM module in the smoothing pre-processing step to improve the overall performance.

2 Related Works

There are many kinds of research about image compression, one of the most traditional ways is to use discrete cosine transform[18] to eliminate the useless high-frequency detail. But when the set bit rate is rather low, distortions are often seen in the final result. In order to eliminate those visual flaw, most of the existing works prefer doing post-processing to remove such artifacts after decompression[1],[2].

Some other works pre-process the input image before compression so that the entropy can be reduced and the artifact can be avoided in the first place[4],[5]. Such technique is not only effective in traditional method, but also in existing work[14], which shows that an image smoothing network before compressing the image improves the compression performance.

During the past several years, there are many work use deep neural network to improve the performance on image compression task[6],[7]. In order to train the compression algorithm as a whole network structure, many works[15],[16],[3] also focus on combining the algorithm into an end-to-end algorithm. Balle et al. proposed jointly training the auto-encoder with the factorized and hyperprior entropy model, respectively and it will be used as a baseline in the following part.

However, all of those DNN-based research focus on the post-processing, so we hope to find a method to use extra feature before the decoding step and get a better rate-distortion performance.

3 The Baseline Approach

This work[13] introduces a deep neural network to traditional JPEG compression technique to make it have better performance on the rate-distortion.

3.1 The JPEG Algorithm

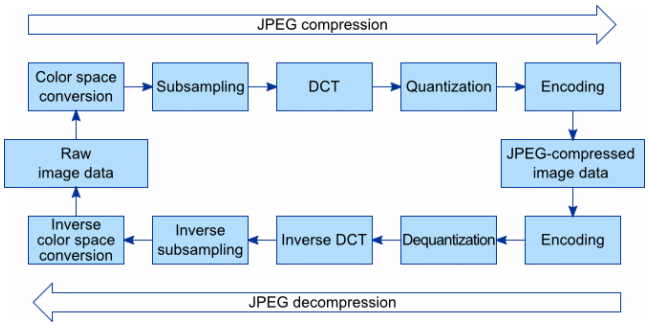

Figure 2. The workflow of traditional JPEG compression (https://www.graphicsmill.com/docs/gm/working-with-jpeg.htm).

The traditional compression method for JPEG is based on the discrete cosine transform. When we get an image waiting for compression, we will transfer it into YCbCr color channel and divide the image into 8x8 pixel groups.

Then we use forward DCT to encode the pixel groups by a combination of the coefficient of a 64 base cosine waves. Then we use the DCT terms we get to divide the quantization table in order to preserve low-frequency information and discard high-frequency (noise-like) detail as humans are less critical to the loss of information in this area, the result will be the compressed result.

Finally, we reverse the previous encode step by using IDCT, unblocking pixel groups then transferring back to RGB channel.

3.2 Attention Network

The baseline paper proposes a novel approach to pre-editing the image before quantization to improve the compression quality. The original paper draws the attention features from the third max pool layer and use a 1x1 convolutional layer with sigmoid to reshape it. The attention tensors for luminance and chrominance for all blocks can be expressed in following way.

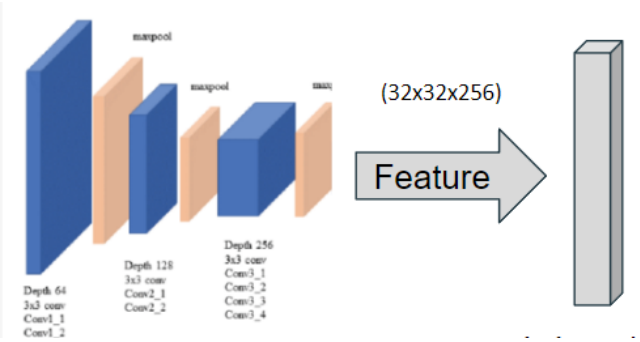

Figure 3. The Attention Mechanism.

$$A^{(L)} \in \{x \in \mathbb{R} | 0 \leq x \leq 1\}^{N \times M \times 8 \times 8} \tag{1}$$

$$A^{(C)} \in \{x \in \mathbb{R} | 0 \leq x \leq 1\}^{N \times M \times 8 \times 8} \tag{2}$$

4 End-to-End JPEG Encoder-Decoder Pipeline

In order to combine the attention features with the traditional compression method and make it can be trained in an end-to-end method, we need to make the compression step differentiable and connect two part together.

4.1 Differentiable Rounding Operation

The traditional rounding operation in the quantization step is indifferentiable, so we cannot combine it to the training process of our network. Therefore, we use a differentiable 3rd order approximation to replace it.

$$\lfloor x \rceil_{approx} = \lfloor x \rceil + (\lfloor x \rceil - x)^3 \tag{3}$$

4.2 Learnable Quantization Table

The JPEG algorithm uses a fixed quantization table to get the compression result, it can only generate relatively stable rate-distortion performance. We use the Hadamard product of the DCT coefficients of block (n, m) for each channel $F[n, m] \in \mathbb{R}^{8\times8}$, the attention tensor $A[n, m]$ and the optimization variables $\mathbf{Q}_\theta^{(L)}$ and $\mathbf{Q}_\theta^{(C)}$ for quantization tables. The formula can be summarized as below,

$$\hat{Z}^{(Y)}[n, m] = \lfloor F^{(Y)}[n, m] \odot A^{(L)}[n, m] \odot \frac{1}{\mathbf{Q}_\theta^{(L)}} \rceil \tag{4}$$

$$\hat{Z}^{(Cr)}[n, m] = \lfloor F^{(Cr)}[n, m] \odot A^{(C)}[n, m] \odot \frac{1}{\mathbf{Q}_\theta^{(C)}} \rceil \tag{5}$$

$$\hat{Z}^{(Y)}[n, m] = \lfloor F^{(Cb)}[n, m] \odot A^{(C)}[n, m] \odot \frac{1}{\mathbf{Q}_\theta^{(C)}} \rceil \tag{6}$$

5 The Proposed Approach

There are many research directions in image compression and some of them demonstrate excellent performance. However, our goal is not to build a state-of-the-art network. Instead, we follow the current structure and propose some potential and practical improvements.

5.1 Add MS-SSIM to Loss

We propose to implement MS-SSIM into the current loss function. Multiscale structural similarity[17] measures the similarity between two images based on luminance, contrast, structure and therefore provides an assessment that is closer to human eyes perception compared with Mean Squared Error (MSE) or Peak Signal to Noise Ratio (PSNR). Therefore, MS-SSIM is a widely used evaluation metric in image compression tasks. The intuition is that MSE only measures absolute difference without considering the actual reflection on human eyes. [20] has proven that a mixture of MS-SSIM and MSE in the loss function could produce more visually appealing images. We follow the similar structure of the loss function as in the baseline paper [13]. The loss function is composed of two parts: distortion loss and rate loss that control image quality and file size respectively:

$$\mathcal{L}(x, \hat{x}; \theta) = \lambda \cdot d(x, \hat{x}) + r(x, \hat{x}; \theta) \tag{7}$$

where $x \in [0, 255]$ is the original image and $\hat{x} \in [0, 255]$ is the compressed image. The weight parameter λ controls the rate-distortion tradeoff. The larger the λ is, the larger the penalty on the distortion loss. Therefore, the image is less compressed. The distortion loss is composed of MSE, Learned Perceptual Image Patch Similarity (LPIPS) which captures difference in perceptual aspects of the image [10], and the proposed MS-SSIM. The new distortion loss function is:

$$\begin{aligned} d(x, \hat{x}) = \text{MSE}(x, \hat{x}) + \gamma \cdot \text{LPIPS}(x, \hat{x}) + \\ \sigma \cdot (1 - \text{MS-SSIM}(x, \hat{x})) \end{aligned} \tag{8}$$

where γ and σ are weight parameters. The MS-SSIM is calculated by the default implementation *tf.image.ssim_multiscale* in TensorFlow. The rate loss is composed of regularized quantized DCT coefficients $\bar{\mathbf{Q}}$ and mean attention maps $\mathbf{A}_\theta^{(L)}$, which stays the same as in the baseline paper:

$$\begin{aligned} \text{r}(\text{x}; \theta) = \alpha \left(\left\| \bar{\mathbf{Q}}_\theta^{(L)} \right\|_1 + \left\| \bar{\mathbf{Q}}_\theta^{(C)} \right\|_1 \right) + \\ \beta \left(\text{mean} \left(\mathbf{A}_\theta^{(L)}(\mathbf{x}) \right) + \text{mean} \left(\mathbf{A}_\theta^{(C)}(\text{x}) \right) \right) \end{aligned} \tag{9}$$

where θ is the learned parameters, α and β are the weight parameters,

5.2 Replace VGG with ResNet

The baseline paper uses VGG-19, a 19 layer variant of the VGG network [12], pre-trained on ImageNet. They extract the image features from the output of the third max-pooling layer that has 256 channels. By the conclusion in [9], ResNet-152 outperforms VGG on both accuracy and speed due to its deep network structure and skip connections. Therefore, it is reasonable to believe that we can extract better features from ResNet. In this paper, we choose a ResNet-152 pre-trained on ImageNet and focus on the third conv layer since it provides relatively good intermediate features. We do experiments on two depths in the third conv larger, which are outputs from block 1 and block 7 respectively. The output from *conv3 block7* is a 28×28 feature map with 512 channels.

5.3 Using LSTMs on Image Feature Maps

In this experiment, we look at the effect of adding LSTMs after the image features are extracted. The idea is to embed sequential information from the feature map into the attention map. The image is divided into 8x8 blocks for DCT. These 8x8 DCT blocks are sequentially fed into LSTMs, one for each channel (Luminance and Chrominance) and the sequence outputs are collected and reshaped into attention map.

Single layer LSTM cells with 64 hidden units are chosen so that the input (8x8 DCT tensor) and the output from the LSTM will be of the same dimension. Thus, the number of time steps is the total number of 8x8 DCT blocks present in the input image. This attention map is multiplied point-wise with the DCT coefficients, as before. **Figure 4** depicts the proposed architecture.

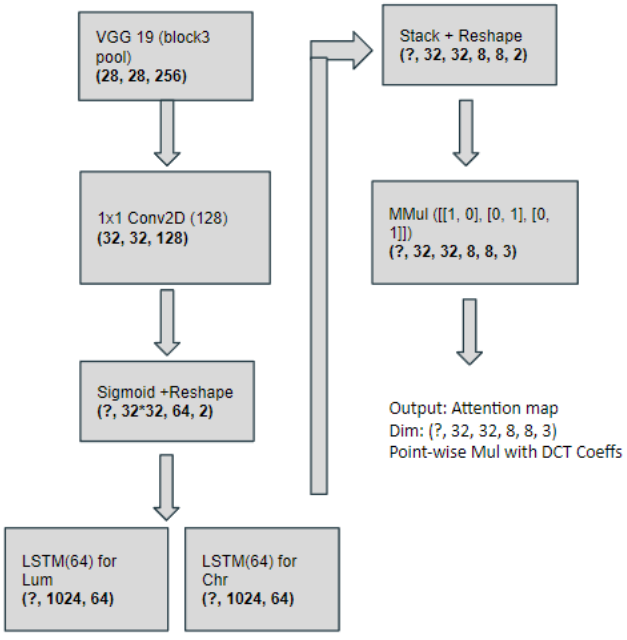

Figure 4. Adding LSTM to the attention network

6 Experiment

6.1 Datasets

For training, we use the HDR+ dataset by Hasinoff *et al* [8]. It contains 3640 HDR images with pixel 4048 × 3036. The preprocess of the training set includes random shuffle of the images, extract training patches of size 224 by random cropping, and random shuffle the patches. For evaluation, we use the Kodak dataset released by the Kodak Corporation for unrestricted research usage (http://r0k.us/graphics/kodak/). It contains 24 lossless, true-color images with resolution of 768 × 512 pixels (24 bits per pixel).

6.2 Training Parameters

The network is trained using Gradient Descent with Adam optimizer. The ResNet and VGG layers of the attention network are initialized using the weights pre-trained on ImageNet. The quantization tables optimization variables are initialized uniformly in the interval [1s, 2s] and are limited to the range [1s, 255s] where s > 0 is a scaling factor. The s is set to $s = 10^{-5}$ due to the small weights from neural network side.[13] The learning rate is 10^{-6}. The patch size is 8. The loss weights as defined in **Equation 8&9** are: $\sigma = 1000$, $\gamma = 500$, $\alpha = 10$, and $\beta = 1$. These are the optimal parameters according to the baseline paper and our experiment. For all experiments, we train 20000 steps.

6.3 Evaluation Metrics

In our experiment, we focus on three evaluation metrics: Mean Squared Error (MSE), Learned Perceptual Image Patch Similarity (LPIPS), and Multiscale Structural Similarity (MS-SSIM). The definition of MSE is:

$$\mathrm{MSE}(x, \hat{x}) = \frac{1}{|\mathcal{P}|} \sum_{\mathrm{p} \in \mathcal{P}} (x_{\mathrm{p}} - \hat{x}_{\mathrm{p}})^2 \tag{10}$$

where x is the original image and $\hat{x}$ is the compressed image and p is a single pixel in the pixel space $\mathcal{P}$. LPIPS, developed by Zhang [19], is another significant perceptual metric that uses deep features to compare similarity. LPIPS takes values in the interval [0, 1] and larger values mean worse image quality. The MS-SSIM is the metric that we focus on the most due to its popularity in image quality. The MS-SSIM is calculated with the default implementation in TensorFlow. MS-SSIM takes values in the interval [0, 1] and larger values mean better image quality. We compare these three metrics across different levels compression rate, which is reflected by bit rate. The bit rate, or Bit Per Pixel (BPP) is calculated by file size divided by total number of pixels.

6.4 Add MS-SSIM to Loss

With the new loss function in **Equation 8**, we compare the performance with the baseline as shown in **Figure 5**. The blue line is the standard JPEG algorithm. The yellow line is the learned JPEG algorithm with VGG attention, which is in the baseline paper. The green line is our approach after adding MS-SSIM into the distortion loss with weight 1000. In MSE, our approach slightly outperforms the baseline at several points. Similar in LPIPS, there is a slight improvement. The most obvious improvement is in MS-SSIM, which makes sense since it is what we add in the loss. As we can see in the middle plot of **Figure 5**, our approach outperforms the baseline at every point in the bpp range 0.25 to 0.42. Without decreasing other metrics, we increase the MS-SSIM, which is considered as the most significant metric in our evaluation. This adding MS-SSIM into the loss function improves the network by generating better compressed images.

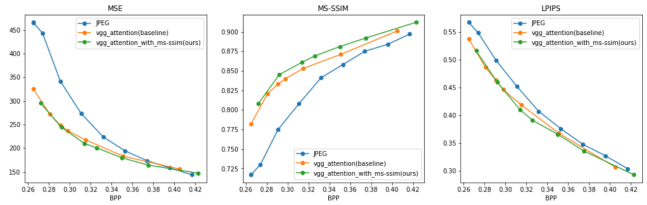

Figure 5. Evaluation of the effect of MS-SSIM in loss

Since the ultimate purpose of image compression is to make the image more visually appealing, looking at the metrics is not enough. In **Figure 6**, from left to right are the original uncompressed image, compressed image (0.315 bpp) by the baseline approach, compressed image (0.314 bpp) by our approach. It is easy to see that our result looks better in terms of color and pixel blocks even with a slightly smaller bit rate. The hair of the middle image is darker than the original image and the pixel block boundaries are unnatural. The area around the nose and eyes is pale ad grey. In the right image, the hair and nose area have more natural color and the pixel blocks are smoother.

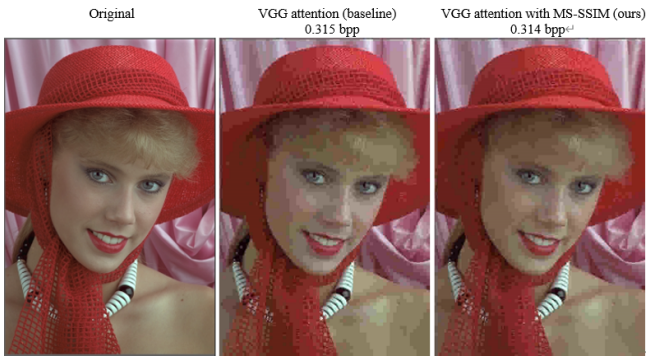

Figure 6. Comparison of compressed images

The VGG attention network with MS-SSIM in loss (the green line) in **Figure 5** is trained with the best weight parameters by our experiment. **Figure 7** shows the comparison of performance with different weight parameters as mentioned

in **Equation 8**. The γ controls LPIPS loss and σ controls MS-SSIM loss. The yellow line is the baseline (without MS-SSIM in loss) and the green line is our optimal weights ($\gamma = 500$, $\sigma = 1000$). The red line ($\gamma = 0$, $\sigma = 1000$) means completely replacing LPIPS with MS-SSIM, which improves the MS-SSIM score without decreasing other metrics. The purple line means both LPIPS and MS-SSIM are included in the loss but with smaller weights. The performance is better than the baseline but not as good as larger weights.

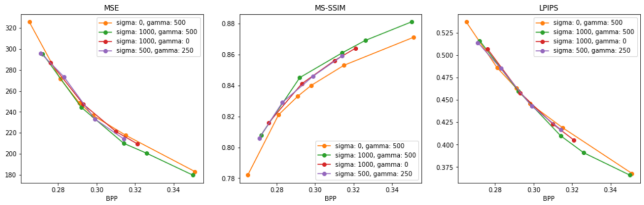

Figure 7. Comparison of different weight parameters

6.5 Replace VGG with ResNet

With the new ResNet based attention network, we compare the performance with the VGG based attention network as shown in **Figure 8**. First, both methods are better than the traditional JPEG method. When take a closer look at the comparison between ResNet and VGG, we can see that they have very similar performance. In MSE, there is one point at 0.315 bpp where ResNet is better than VGG. In other points, they are almost identical. Similar pattern shows in LPIPS. IN MS-SSIM, the pattern is different. The score of ResNet is lower than VGG in bpp interval [0.26, 0.31].

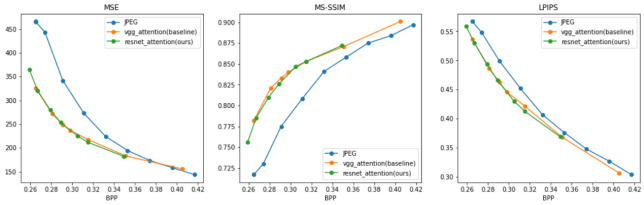

Figure 8. Evalutaion of ResNet and VGG based attention network

Summarize all three metrics and consider that MS-SSIM is the most significant metric in our evaluation, we can not say that ResNet based attention network is better than VGG in compression performance. The same conclusion could be drawn from **Figure 9**. The compressed images show no obvious difference in raw eyes at the same bit rate.

However, notice that in **Figure 8** we compare the ResNet and VGG based attention network without including MS-SSIM in the loss function. In **Figure 10**, we compare them after adding MS-SSIM into the loss function. We can see that after adding MS-SSIM, ResNet performs better than VGG in

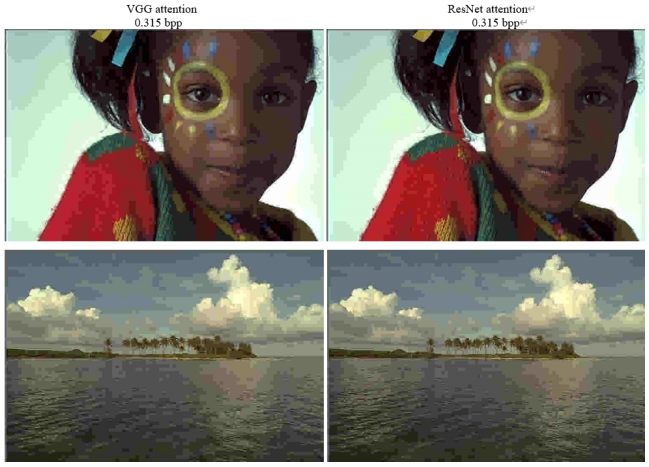

Figure 9. Compressed images from ResNet and VGG based attention network

the bpp interval [0.3 to 0.38]. This means by combining the two proposed changes in Section 5.1 and 5.2, we get a better overall performance than the two changes alone.

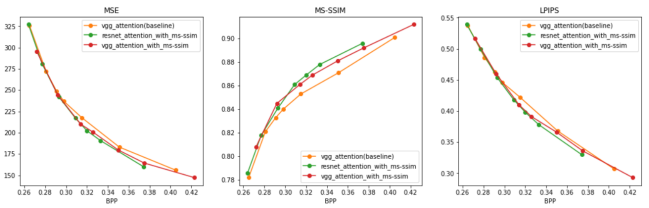

Figure 10. Evaluation of ResNet and VGG based attention network with MS-SSIM

Another thing to notice is that the training speed of ResNet is faster than VGG. In **Figure 11** generated by TensorBoard, with the same settings, the speed of ResNet is about 57% higher than VGG. This could be another reason we choose ResNet when we have a larger training set and training steps.

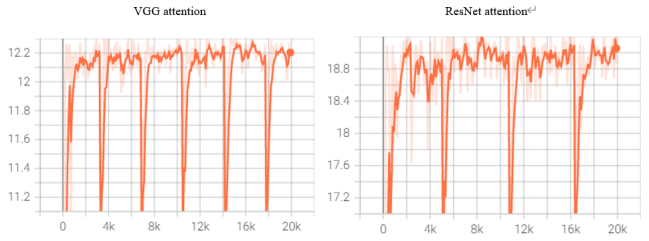

Figure 11. Training speed (steps/sec) of ResNet and VGG based attention network

6.6 Runtime Analysis

First, we compare our inference time with the end-to-end model which is proposed by Ballé[3],

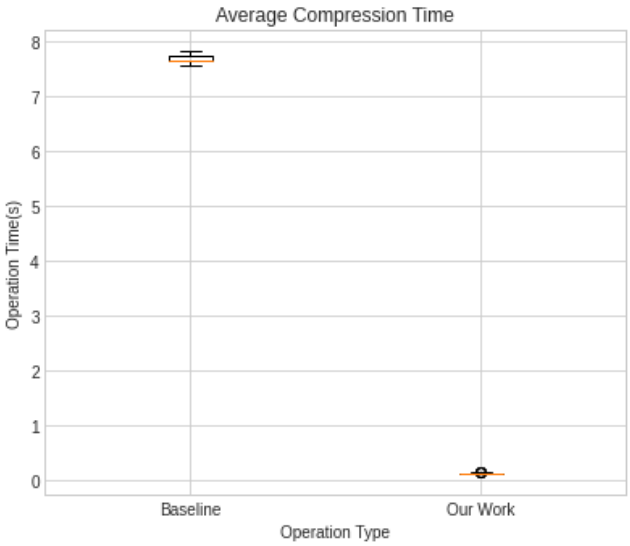

Figure 12. The Inference Time Comparison

The mean inference time of our model is 0.126 with the standard error 0.010 while baseline has a mean inference time at 7.667 with the standard error 0.079. We can find that our model has far less inference time than the model of baseline as well as more stability.

6.7 Metrics against Training Steps

Then we test the performance of model with and without attention mechanism against the training step, in order to demonstrate the metrics in the same figure, the metrics have been normalized.

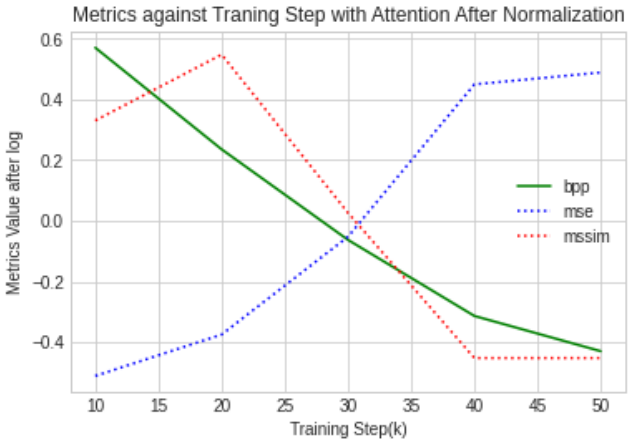

Figure 13. The metrics change when the training steps increase with attention mechanism

With the training step increases, the bpp decrease means the compression rate increase, it will increase the mean square error but reduce the MS-SSIM metric value.

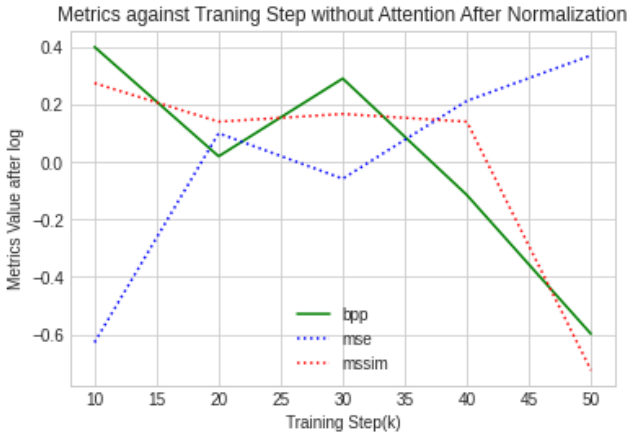

Figure 14. The metrics change when the training steps increase without attention mechanism

6.8 Evaluation of LSTMs in attention map

The LSTM based architecture described above is implemented. Learning rate of 1e-6 is chosen based on initial analysis, where it is observed that any LR greater than 1e-6 leads to over-shooting and hence, leads to unreliable learning over steps. This is in accordance with the work of Yannick et al [13]. **Figure 15** shows how LSTM and Bi-directional LSTM based attention perform against vanilla CNN based soft-attention. *BRNN* is the Bi-directional LSTM based model, *attn-LSTM* is the LSTM version of the former and *vanAttention* is the CNN only attention model proposed by Yannick et al.

We can observe that LSTM and Bi-LSTM based models perform sub-optimally as compared to their CNN-only counterpart. We have tried normalizing the LSTM based attention map, using min-max normalization, before multiplying it with DCT coefficients, this resulted in further degrading the performance. The incorporation of Bi-LSTM is based on the idea that processing patches of images not only from past-to-future but also from future-to-past might add better sequential information to the attention map. Unfortunately, this idea leads to performance lower than a single LSTM layer.

As the BPP i.e. the amount of information or bits per pixel is increasing, the MS-SSIM saturates meaning that the RNN modules are not able to optimally decide what to forget and what to remember, assuming that the sequential information in the image feature map is still pertinent to compression. But, if the sequential information is indeed important, it must show this at the lower BPP levels where these models still are struggling to beat the vanilla JPEG baseline. This leads to a strong argument that some other idea is to be pursued for deep pre-editing the image before compression. This compels us to look into methods such as iterative refinement, as presented in[11].

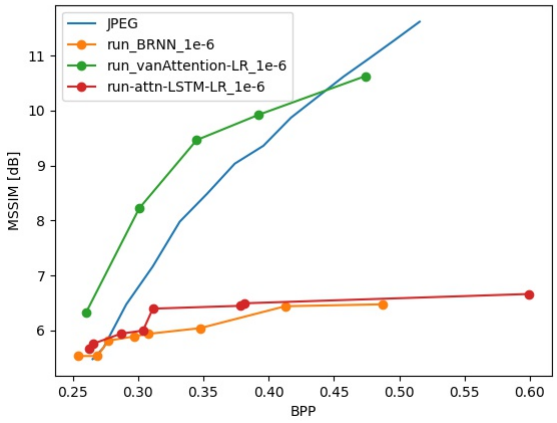

Figure 15. MS-SSIM vs BPP for LSTM based attention

7 Conclusion

In this paper, we analyze the work by Strumpler et al[13] and propose improvement based on it. The added MS-SSIM loss term is proven to be useful as it generates better image quality. Although ResNet performs slightly worse than the VGG based attention network without MS-SSIM, it is better after considering MS-SSIM. We successfully improve the baseline when combing the proposed approaches. We also analyze the training speed and inference time. Unfortunately, the implemented LSTM does not demonstrate a good performance. We will keep looking into it for better implementation if we have more time.

References

[1] François Alter, Sylvain Durand, and Jacques Froment. 2005. Adapted Total Variation for Artifact Free Decompression of JPEG Images. *J. Math. Imaging Vis.* 23, 2 (sep 2005), 199–211. https://doi.org/10.1007/s10851-005-6467-9

[2] A. Averbuch, Alon Schclar, and D. Donoho. 2005. Deblocking of block-transform compressed images using weighted sums of symmetrically aligned pixels. *Image Processing, IEEE Transactions on* 14 (03 2005), 200 – 212. https://doi.org/10.1109/TIP.2004.840688

[3] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. 2016. End-to-end Optimized Image Compression. (2016). https://doi.org/10.48550/ARXIV.1611.01704

[4] Yehuda Dar, Michael Elad, and Alfred Bruckstein. 2017. Optimized Pre-Compensating Compression. *IEEE Transactions on Image Processing* PP (11 2017). https://doi.org/10.1109/TIP.2018.2845125

[5] Yehuda Dar, Michael Elad, and Alfred Marcel Bruckstein. 2018. System-Aware Compression. *2018 IEEE International Symposium on Information Theory (ISIT)* (2018), 2226–2230.

[6] Chao Dong, Yubin Deng, Chen Change Loy, and Xiaoou Tang. 2015. Compression Artifacts Reduction by a Deep Convolutional Network. (04 2015). https://doi.org/10.1109/ICCV.2015.73

[7] Jun Guo and Hongyang Chao. 2016. Building Dual-Domain Representations for Compression Artifacts Reduction. In *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, Cham, 628–644.

[8] Samuel W Hasinoff, Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan T Barron, Florian Kainz, Jiawen Chen, and Marc Levoy. 2016. Burst photography for high dynamic range and low-light imaging on mobile cameras. *ACM Trans. Graph.* 35, 6 (Nov. 2016), 1–12.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[10] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. 2016. Photo-realistic single image super-resolution using a generative adversarial network. (Sept. 2016). arXiv:1609.04802

[11] Ankur Mali, Alexander Ororbia, Daniel Kifer, and Lee Giles. 2022. Neural JPEG: End-to-End Image Compression Leveraging a Standard JPEG Encoder-Decoder. https://doi.org/10.48550/ARXIV.2201.11795

[12] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. (Sept. 2014). arXiv:1409.1556

[13] Yannick Strümpler, Ren Yang, and Radu Timofte. 2020. Learning to Improve Image Compression without Changing the Standard Decoder. (09 2020).

[14] Hossein Talebi, Damien Kelly, Xiyang Luo, Ignacio Garcia Dorado, Feng Yang, Peyman Milanfar, and Michael Elad. 2021. Better Compression With Deep Pre-Editing. *IEEE Transactions on Image Processing* 30 (2021), 6673–6685. https://doi.org/10.1109/tip.2021.3096085

[15] George Toderici, Sean O'Malley, Sung Hwang, Damien Vincent, David Minnen, Shumeet Baluja, Michele Covell, and Rahul Sukthankar. 2015. Variable Rate Image Compression with Recurrent Neural Networks. (11 2015).

[16] George Toderici, Damien Vincent, Nick Johnston, Sung Hwang, David Minnen, Joel Shor, and Michele Covell. 2016. Full Resolution Image Compression with Recurrent Neural Networks. (08 2016).

[17] Z Wang, E P Simoncelli, and A C Bovik. 2004. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003* (Pacific Grove, CA, USA). IEEE.

[18] Andrew Watson. 1994. Image Compression Using the Discrete Cosine Transform. *Mathematica Journal* 4 (08 1994). https://doi.org/10.1007/978-3-322-96658-2_5

[19] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (Salt Lake City, UT). IEEE.

[20] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. 2017. Loss functions for image restoration with neural networks. *IEEE Trans. Comput. Imaging* 3, 1 (March 2017), 47–57.