Image Compression using Hierarchical Modeling

Alekhya Pyla; Project Partner - Prakhar Srivastava

1 Project Overview:

As the internet reaches billions of people worldwide, data streaming speed becomes highly important. Data compression techniques aim to reduce the time and bandwidth costs, making information flow quickly across networks and reducing the space consumed. Image compression is a method to remove spatial redundancy between adjacent pixels and have the ability to reconstruct a high-quality image. In the past few years, machine learning has been successful in producing promising image reconstruction results. The traditional image compression methods like JPEG use linear data transformations for compression but fall behind to maintain quality with low bits per pixel and it suffers from the severe artifacts commonly seen in linear transform coding methods.

With neural networks, we can instead perform non-linear transform coding, to get better quality compressed images by trying to improve the key performance metric which is the rate-distortion tradeoff. The compression methods compared in this work are all lossy compressions where some information is lost during the process as opposed to lossless compression. Rate is the number of bits required to compress the image and distortion is the deviation of decompressed image (from the bitstring) from the ground truth. Usually, low rates mean higher distortion and vice-versa. Transform coding consists of an analysis and synthesis steps that perform the compression and decompression respectively. We have a set of Variational Auto-Encoder model architectures, built using convolutional neural networks for each of them both. We jointly optimize the entire model for rate-distortion performance by training over a set of images. The inspiration for this work is derived from Balle et al (2017) (1). We aim to use this as a performance benchmark, which most of the report revolves around. To improve upon this, we aim to construct hierarchical networks, with multiple layers of latent space for compression, which my project partner worked upon. The reason for using hierarchical networks is to leverage the hyper-priors in the entropy model.

2 Approach:

2.1 Transform coding :

Transform coding is a general image compression technique that involves three main steps- Transform, Quantize, and Encode (2) as shown in Figure 1. In the first step, Transform converts the image to the higher-order domain(like the frequency domain). A fewer number of coefficients try to capture most information of the image. To map the pixel value of the

image to a series of coefficients, a reversible linear transformation, such as Discrete Fourier or Discrete Cosine Transform is used in compression techniques like JPEG. These coefficients are quantified and coded further. For a high-quality transformation, higher information is captured within a smaller number of coefficients.

As the next step, the quantization technique filters coefficients that contain the most information needed. It maps many to one, thereby reducing the number of bits needed to represent the image. For example, an image of dimensions NxN is broken down into smaller blocks of nxn and each block is mapped to a single value.

Finally, the result is encoded using entropy coding to further improve the compression. In information theory, entropy coding (or entropy encoding) is a lossless data compression scheme that is independent of the specific characteristics of the medium. One of the main types of entropy coding is where it creates and assigns a unique prefix-free code to each unique symbol present in the input. The data is compressed by replacing each input symbol with the prefix-free output codeword. The length of each codeword is approximately proportional to the negative logarithm of the probability of occurrence of that codeword. Therefore, the most common symbols use the shortest codes. change to your explanation. Entropy is the lowest number of bits required to represent a symbol in the compression frame (the total length of the code for all symbols).

On the other hand, for decompression, which is to reconstruct the original image, similar procedures in opposite order exist; decoding and inverse transformation. Transform coding is a lossy compression technique.

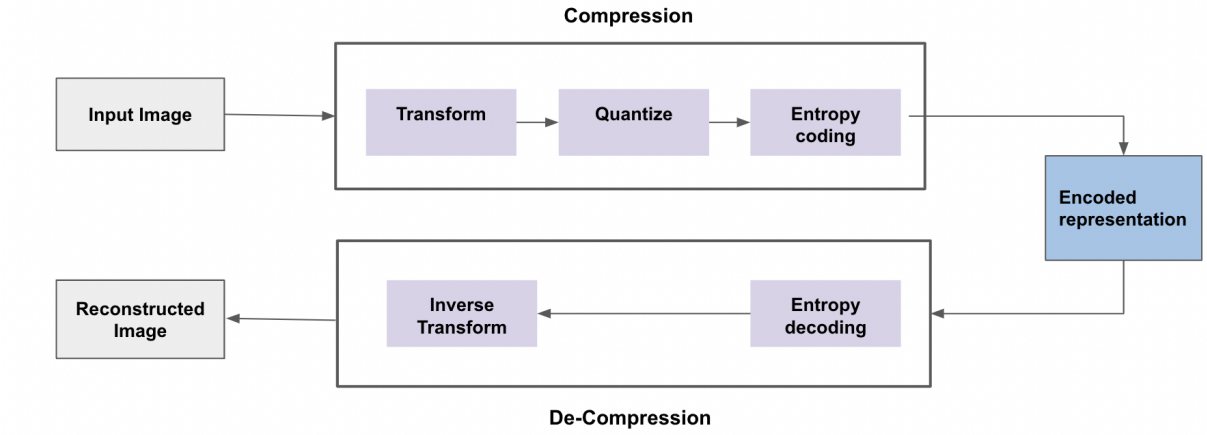


Figure 1: Transform coding framework.

2.2 Compression using Convolution Neural Networks:

Instead of using linear transformations, Neural networks can be replaced for non-linear transformation coding of the images. In this case, end-to-end methods of compression and de-

compression are done using two different neural networks. In our work, multiple layers of Convolutional Neural Networks (CNNs) were used for both models as shown in Figure 2. CNN's were found successful for computer vision tasks like classifying images, primarily due to their ability to learn features via convolution, that is by identifying the dependencies between neighboring pixels. A general convolution neural network consists of input and hidden layers, consisting of convolution, pooling, fully connected, and normalization layers. The shared-weight architecture of the convolution kernels or filters that slide along input features provide translation-equivariant responses known as feature maps. This non-linear transform coding for compression and decompression is called Synthesis and Analysis Transform.

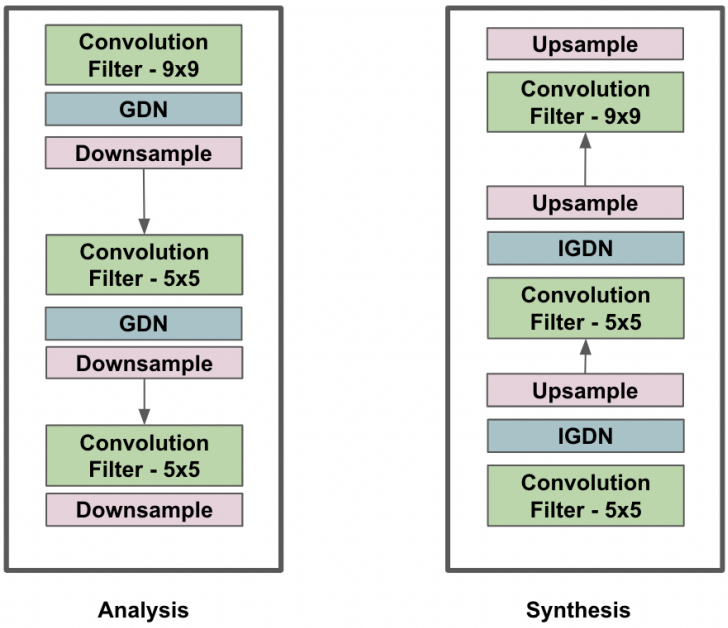


Figure 2: Architecture

2.2.1 Analysis Transform

Figure 3 shows the transformations with CNNs used to generate rate and distortion. The analysis transform, $\boldsymbol{y} = \boldsymbol{g_a}(\boldsymbol{x};\boldsymbol{\phi})$, represents the encoder CNN, which takes image intensities X as input and maps it to a latent space, Y. These latent vectors are quantized, giving a discrete representation Y. One of the main challenges of using neural networks for compression is that the latent space values generated from the networks are continuous. During the quantization step, these continuous latent space vectors are converted to discrete values. A simple quantization model (3) would be to have y = q = round(y). Post quantization, the derivatives become zero almost and it makes the gradient descent not possible to compute. To allow optimization via stochastic gradient descent, a uniform noise is added, $\tilde{\boldsymbol{y}} = \boldsymbol{y} + \boldsymbol{\Delta y}$ is the continuous relaxation of the probability mass function of q , where $q \in Z$. It is assumed that the quantization bin size is always one and the representing values are at the centers of

the bins as shown :

$$P_{\boldsymbol{q}_i}(n) = \int_{n-\frac{1}{2}}^{n+\frac{1}{2}} p_{\boldsymbol{y}_i}(t)\, dt \tag{1}$$

The data is sampled from this distribution for quantized values and the same approximation is used to optimize the loss. The quantization introduces errors, thereby leading to the rate-distortion optimization. Rate, R is the compressed image representation in bits. R represents the average bits needed to represent one pixel. Using entropy encoding techniques, the Rate is calculated using cross-entropy from the equation below where Q is the quantization function applied :

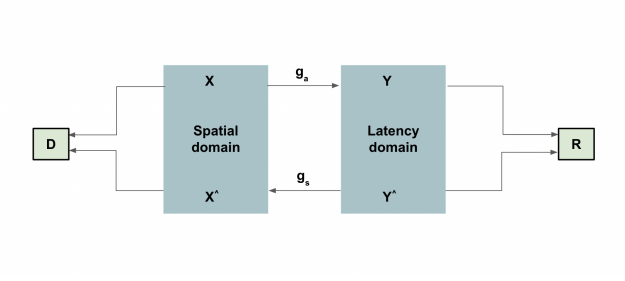


Figure 3: Synthesis and Analysis Transforms

$$R = \mathbb{E}_{\boldsymbol{x}\sim p_x}\left[-\log_2 p_{\hat{\boldsymbol{y}}}\left(Q\left(g_a\left(\boldsymbol{x};\boldsymbol{\phi}_g\right)\right)\right)\right] \tag{2}$$

Also, R can be stated as the number of bits/ number of pixels.

2.2.2 Entropy Encoding

In the implementation, the Continuous-Batched entropy encoding technique that uses arithmetic encoding in the background is applied to generate R. The entropy technique uses a Noisy Deep Factorized prior, to help estimate the probability distribution of the latent space for calculation of R. The fully factorized takes a marginal distribution, ignoring dependencies within dimensions. It can be observed from Figure 4, that the latent space post-Deep Factorized, which has some statistics of data remained. This doesn't give the optimal entropy possible. To address this issue, scaled hyperpriors (4) are introduced using hierarchical structures in the models. A hyper-synthesis and hyper-analysis transform are added to the network as shown in Figure 5.

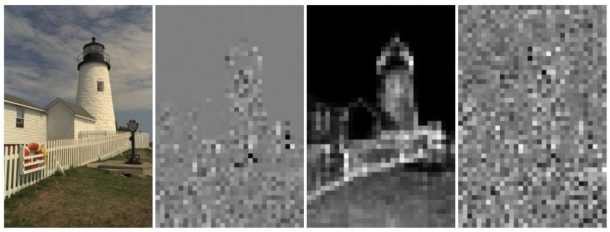


Figure 4: Latent Space with Noisy Deep Factorized prior on the left and with hyperprior on the right

Each element $\hat{y}$ is now modeled as a zero-mean Gaussian with its standard deviation sigma, predicted by a hyper-synthesis transform h_s. In this project, we extend the hyperprior approach to multiple hierarchical levels, where the topmost level is the hyperprior that dictates the standard deviation of each latent variable in the hierarchies below it. Note that it has been convolved with the standard uniform to better fit the target distribution, commonly referred to as the Noisy Normal. On the encoder side of things, a hyper-analysis transform h_s is simply stacked over y to get z. Since we have no prior belief about the hyperprior, we can simply model it with the Noisy Deep Factorized model.

2.2.3 Sythesis and Loss Optimization

For the synthesis transform, $\boldsymbol{x} = \boldsymbol{g_s}(\boldsymbol{y};\boldsymbol{\theta})$, , the discrete elements in latent space are used to reconstruct the original image. Distortion is calculated by taking the mean squared error between the input and reconstructed data. We optimize the parameter vectors ϕ and θ, of analysis and synthesis CNNs by taking a loss function of the combination of R and D :

$$\mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}}} D_{KL}\left[q \| p_{\tilde{\boldsymbol{y}}|\boldsymbol{x}}\right] = \mathbb{E}_{\boldsymbol{x}\sim p_{\boldsymbol{x}}} \mathbb{E}_{\tilde{\boldsymbol{y}}\sim q}\Bigg[\cancelto{0}{\log q\left(\tilde{\boldsymbol{y}} \mid \boldsymbol{x}\right)} \underbrace{-\log p_{\boldsymbol{x}|\tilde{\boldsymbol{y}}}\left(\boldsymbol{x} \mid \tilde{\boldsymbol{y}}\right)}_{\text{weighted distortion}} \underbrace{-\log p_{\tilde{\boldsymbol{y}}}\left(\tilde{\boldsymbol{y}}\right)}_{\text{rate}}\Bigg] + \text{const.} \tag{3}$$

$$\begin{aligned} Loss &= R + \lambda * D \\ D &= MeanSquaredError(x, x) \end{aligned} \tag{4}$$

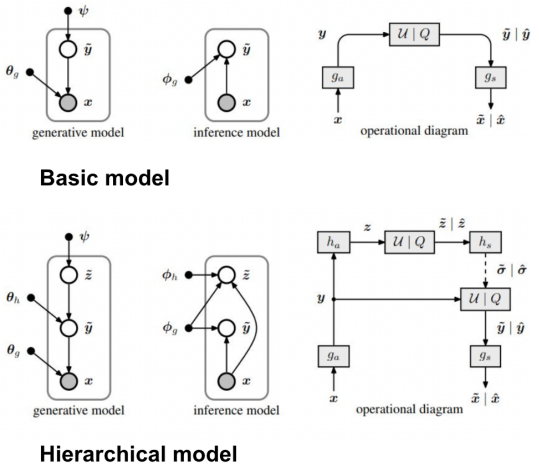


Figure 5: Basic model and Hierarchial model with hyper-priors

This loss function can also be interpreted from a variational inference point of view as optimizing the KL divergence as shown in equation 3. The first term is a uniform distribution that is centered around y. Since the width of this distribution is constant (one), the first term in the optimization equation turns out to be zero. The remaining two terms represent distortion and rate.

The optimization is done over training images for a varying set of lambda values. The loss function is inspired by the lagrangian optimization problem. Distortion represents a loss of image quality, and rate represents the amount of data required to encode the image, These two metrics work against each other, that is when distortion will be low, the rate would be high, and vice versa. The decisions have to be made that affect both file size and quality simultaneously, depending on the task at hand.

3 DataSet:

The dataset used is Coco(Common Objects in Context) 2017 Dataset, which was also used for large-scale object detection, segmentation, and captioning dataset. The dataset consists of 164K images. In the 2017 version used, the training, validation, and test split is 118K, 5K, and 41K respectively. This dataset is publicly available for download. Some examples of images are shown in Figure 6.

4 Implementation Details :

Figure 6, shows the architecture of the network. The models are built using the Tensorflow compression module TFC. There was a starter code available to help process the data and adjust configurations. The majority of work involved building the models on top of the starter code. The Analysis transform consists of three layers of CNN, with filter sizes as shown in the image. The first two layers of CNN have GDN activation enabled. Generalized Divisive Normalization (GDN) (5), inspired by models of neurons in biological visual systems, has shown the best performance for compression tasks, hence used in the analysis model.

Figure 6: Sample Images

Similarly, IGDN (Inverse GDN) is used for the synthesis model as an activation function. The downsampling is done by setting the stride_down to values of 4,2,2, which indicates that dimensions are reduced to 1/4th, and 1/2th in the layers respectively. Similarly, the Synthesis transform also consists of three layers of CNN, with upsampling(stride_up is set to be 2,2, and 4), to get back to the original image dimensions.

A Continuous Batched entropy Model from TensorFlow is used for handling the quantization of the bottleneck tensor (the output of the analysis or the latent representation between analysis and synthesis models).

For the hierarchical models, 6-layer deep hierarchical VAE which follows bidirectional inference is built. ResNet blocks are used to construct this instead of the simple Convolution blocks (6). The topmost encoder-decoder use the Noisy Normal hyper-prior model for encoding as described previously and the remaining lower hierarchical models use Noisy Deep Factorized prior. The architecture remains the same in terms of activation function as before, with major differences in the presence of skip-connections between multiple layers of encoder-decoder blocks.

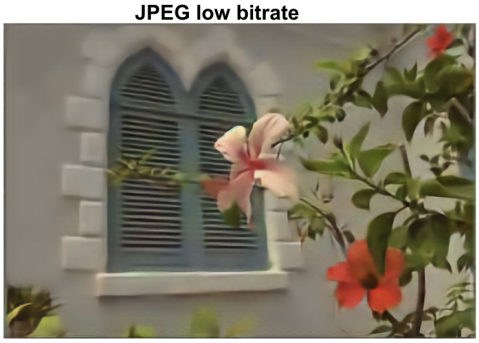
The CNN model was trained for 200 epochs whereas the hierarchical one was trained for 25 epochs for varying lambda values. The lambda values used are 8.02, 2.29, 0.65, 0.19, 0.05, 0.015, 0.004, 0.001, 0.0006, which are picked randomly.

5 Results

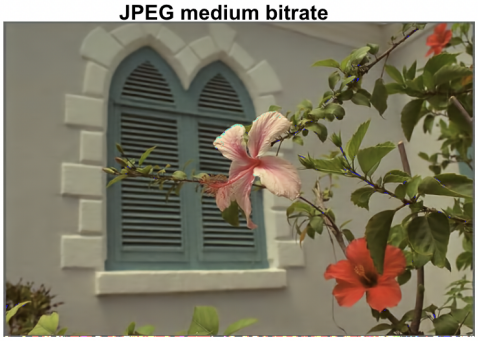
JPEG low bitrate

JPEG medium bitrate

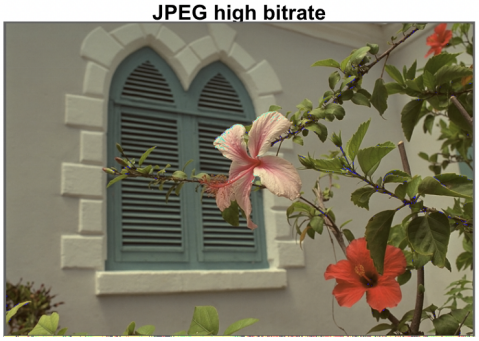
JPEG high bitrate



BLS low bitrate



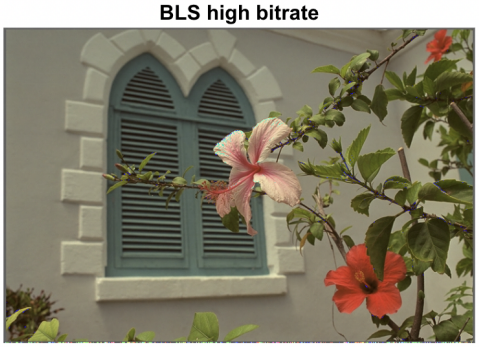
BLS medium bitrate



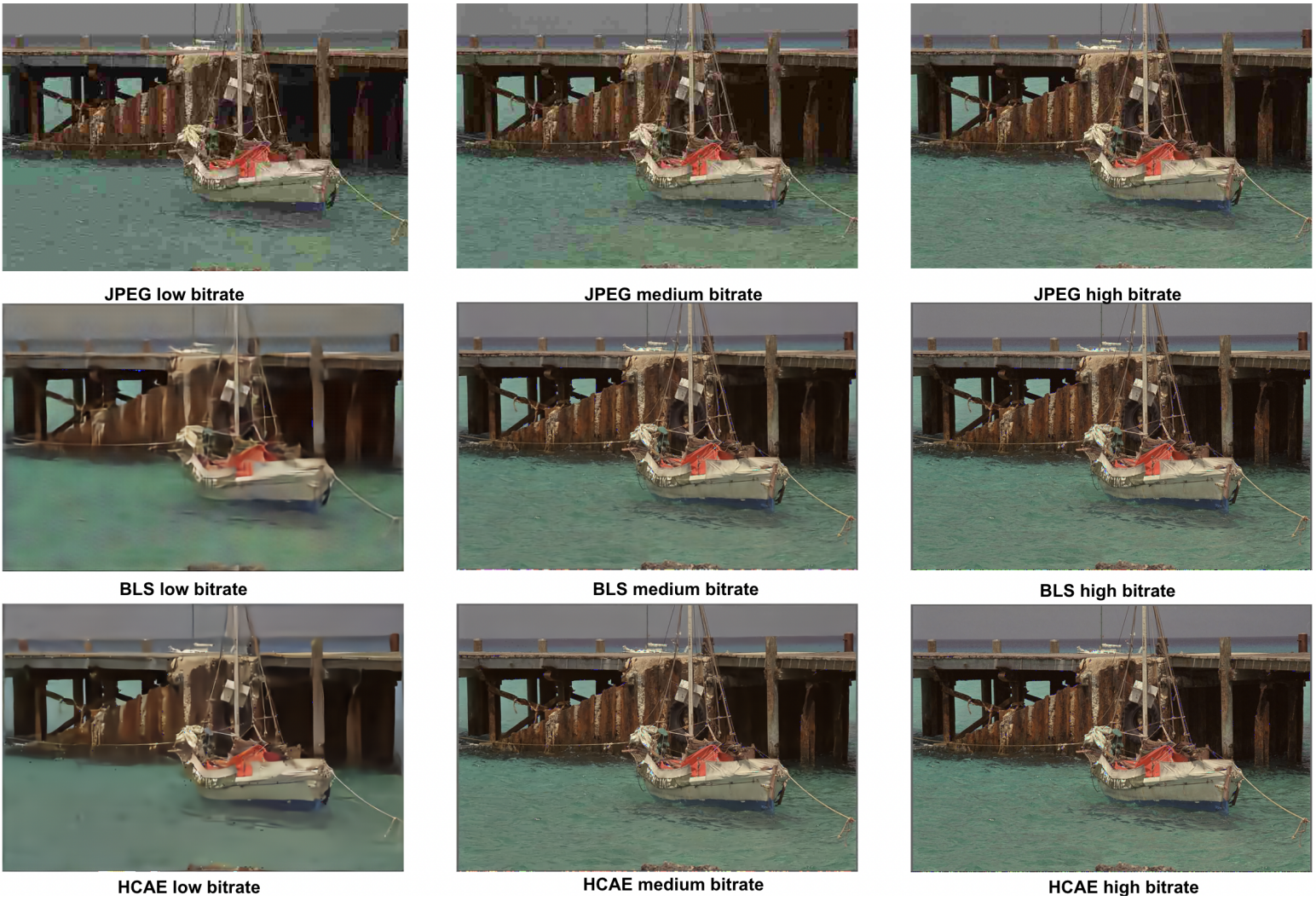
BLS high bitrate

HCAE low bitrate

HCAE medium bitrate



HCAE high bitrate



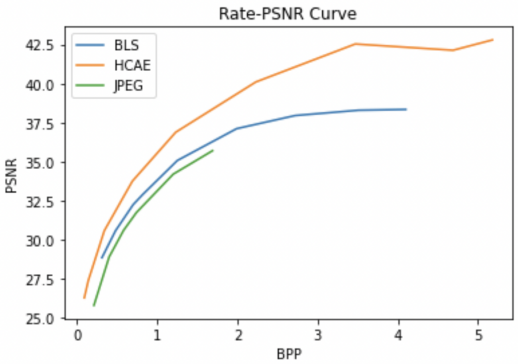


Figure 7: Rate-PSNR

The results are shown for a set of three different bit rate regimes - low, medium, and high across the JPEG, CNN VAE model also referred to as BLS and Hierarchical model also

referred to as HCAE. The vast improvement in image quality can be found in the low-bit-rate regime. From the attached images, it is observed that for a low bitrate, the JPEG images contain a lot of artifacts. The bad image quality is widely improved upon using the BLS model (Auto-Encoder CNN) or Hierarchical (HCAE) model and the artifacts are not visible in either of them. Similar quality improvement is also visible in the medium bit rate. For the high bit rate, it is difficult to spot stark differences qualitatively. The change is measured quantitatively using the Rate vs PSNR curve shown in Figure 7, where BPP stands for Bits per pixel.

PSNR stands for Peak signal-to-noise ratio, which is an expression for the ratio between the maximum possible value (power) of a signal and the power of distorting noise that affects the quality of its representation. In simple terms, PSNR captures the inverse of distortion, i.e good quality images have high PSNR. It is observed that the basic VAE model(BLS) has high PSNR than JPEG for the same bitrate. Also, the Hierarchical model (HCAE) performs better than the BLS model and JPEG. Thus, for a particular bit rate, HCAE performs best, followed by the CNN AutoEncoder model and then JPEG.

6 Assessment and Evaluation:

The task was fairly difficult for the computer algorithm to solve considering the vast information theory background that is needed to understand the task at hand. Optimizing for the entropy model and distortion jointly is also a challenge over the high dimensional latent space. The results were limited due to the high computational requirement needed. A smoother rate-distortion performance curve could be possible if the training was done for more values of lambda. This approach could be prone to high errors in testing in out-of-distribution images, leading to artifacts in the compressed images. Also, when tested on images of the different datasets but in the domain, artifacts were found. The success of the approach is that the method gave an optimized rate-distortion performance. Also, although the method used Mean Squared Error for distortion in loss optimization, the compressed images are much more natural in appearance than JPEG which suffers from severe artifacts in the reconstructed images. We believe this visual improvement arises because biologically-inspired nonlinear transformations in the model might have been better performant to capture and eliminate the components which the human eye cannot detect and thus giving compressed images with good quality. Further, it is observed that the hierarchical model performed better due to efficient modeling of the entropy to generate lower bit rate and distortion. A better system can be built by replacing the convolutional blocks in the architecture with Vision transformers.

References

[1] Ballé J, Laparra V, Simoncelli EP. End-to-end optimized image compression. arXiv preprint arXiv:161101704. 2016;.

[2] Sadeeq H, Hameed T, Abdi A, Nashwan A. Image Compression Using Neural Networks:

A Review. International Journal of Online and Biomedical Engineering (iJOE). 2021 12;17:135–153.

[3] Gray RM, Neuhoff DL. Quantization. IEEE transactions on information theory. 1998;44(6):2325–2383.

[4] Ballé J, Minnen D, Singh S, Hwang SJ, Johnston N. Variational image compression with a scale hyperprior. arXiv preprint arXiv:180201436. 2018;.

[5] Ball J, Laparra V, Simoncelli EP. Density Modeling of Images using a Generalized Normalization Transformation. 2015;Available from: https://arxiv.org/abs/1511.06281.

[6] Kingma DP, Salimans T, Jozefowicz R, Chen X, Sutskever I, Welling M. Improved variational inference with inverse autoregressive flow. Advances in neural information processing systems. 2016;29.

7 Appendix

The starter code, a pipeline code to run the models, was available from one of our previous projects which we had reused to enable this project. The reused code is boilerplate.py, keras_utils.py, models_utils.py. We have implemented from scratch the bls2017.py, hcae.py, and some of the utils.py. The utils.py was modified to fit the needs of the project.