Faster Image Compression Using Network Compression

Riley Johnson
Northeastern University
360 Huntington Ave
Boston Ma, 02115
johnson.k@husky.neu.edu

Raghav Avasthi
Northeastern University
360 Huntington Ave
Boston Ma, 02115
avasthi.r@husky.neu.edu

Abstract

Mobile devices have allowed people to carry high-resolution cameras and the computational power to process them around at all times. The ability for mobile devices to run inference on the images stored on them is limited by the resources that can fit in a small device. Memory, battery life, and storage space are all limiting factors in the tasks that can be performed. For this, we propose the application of LZMA encoding to an image compressing autoencoder which would decrease the execution time of a state-of-the-art image compressing autoencoder by multiple folds making it viable for practical purposes on resource-constrained devices such as mobiles, with a little tolerable loss in compressed image quality and compression rate. Specifically, we made use of these compression methods such as encoding and quantization to enable the task of image folder compression, one that is often necessary for devices with not only limited storage but limited data transfer capacity due to mobile networks.

1. Introduction

With the rise of mobile devices as a means of processing data it has become necessary to scale things down. Memory, power consumption, storage, processing power, and data transfer are all sacrificed for the convenience of having a portable device with a wide variety of features. Mobile devices are more and more frequently bragging better and better cameras, offering the ability to capture high resolution images of the world around us. Megapixel counts and lens specifications are advertised front and center. But storing and sending high resolution images can be inconvenient.

1.1. Image compression

State of the art deep learning techniques have led to new advancements in image compression. It is now possible to achieve lower bits per pixel with high structural similarity by breaking away from traditional encoding methods and allowing a network to make these decisions for us.

Auto encoders are an intuitive solution to efficient image representation. An image is transformed into a comparatively small feature vector and the decoder network is trained to restore the pixel data. These types of network are often simple and end-to-end trainable, and in the case of image compression it should use an unsupervised loss function. These make auto encoders an attractive method for dealing with image compression.

Another approach is the use of generative adversarial networks. The idea behind the use of an adversarial loss for this task is that instead of training the network based on structural similarity between the input image and the reconstructed image, a discriminator can be used to make decisions based on how close the reconstruction is to the original, leading to outputs that are very structurally and stylistically similar to the input and with rich color and sharp edges. However, these types of networks are difficult to train and even more difficult to validate, and more complex network architectures like this can be difficult to apply network compression on.

1.2. Compression techniques

The performance of deep neural networks is often limited by the amount of memory needed to store the parameters as well as the amount of time it takes to perform floating point operations. Thankfully, various techniques have been introduced to reduce the scale of networks without sacrificing accuracy.

Pruning can increase network efficiency by removing entire convolutional filters, thereby reducing the number of parameters and operations in the network. This is an iterative process that involves removing filters that contribute little to the output, removing them, and then fine tuning the network to recover its accuracy.

Quantization reduces the number of bits used to stores parameters and simplifies the mathematical operations performed by the network by converting large floating point numbers to discrete integer values. This works by assigning weights to the nearest integer value and then fine tuning the network to recover its accuracy.

After removing weights and reducing the number of bits needed to store them the next step is Huffman coding.

Huffman coding is an encoding method that benefits from similar weights being present in the network. In fact, many networks will have a clear bias in the distributions of their symbols, allowing Huffman coding to drastically reduce the network size.

1.3. Unified techniques

The idea for combining these two techniques is to create a system for storing high resolution images that is fully geared towards mobile systems. The output of the compression network is a small representation of the image that can be reconstructed with high perceptual accuracy. The network itself should be small, taking up little memory and operating quickly. Ideal for mobile use.

2. Related work

A great deal of work has been done in both the field of image compression as well as network compression. Both of these concepts must be explored in order to merge the state of the art for these methods into a unified implementation for mobile image compression.

2.1. Image compression overview

The most intuitive method for compression images using neural nets involves using image auto-encoders, either convolutional as in [2, 6, 7] or through the use of recurrent neural networks as in [3, 8]. This method compresses an image into a low dimensional feature space and trains the network by its ability to reconstruct the original image accurately. The decoder network is kept and the bitrate of the image is determined by the "bottleneck" layer between the encoder and decoder networks. Effectively making bitrate tied to a design parameter and allowing accuracy to be optimized.

How accuracy is determined is involves selecting the right metric. L2 loss is the easiest metric to implement but it doesn't align well with human perception. When running a lossy compression algorithm information is going to be lost, but choosing the right accuracy measure can ensure that information is kept where it matters. This is where metrics like structural similarity become useful, modeling reconstructed image accuracy in a way that values the image structure as a human would perceive it. An additional method of determining accuracy is deformation aware accuracy which finds which deformations of the image allow it to encode more efficiently without heavily affecting the overall structure of the image [1].

Progress has been made recently with generative adversarial networks for image compression as well [3, 4]. This approach is a dramatic shift from previous approaches in that instead of optimizing some specific metric of accuracy between the reconstructed image and the input image, it uses a discriminator network to determine if an image is real or reconstructed. When trained correctly this can lead to visually impressive results with record low bitrates. This method is rather difficult to train however, and the network itself is rather large and complex.

Despite the visually impressive results offered by adversarial losses for this task it strays away from our design goal of having a simple, easily prune-able network, with few weights. The abilities of some auto-encoder based solutions are capable of providing the desired compression rates while still looking good enough when run on the Kodak image set.

2.2. Image encoding

Lossy image compression has been a long-standing problem in the field of image processing and computer vision. There are many standard codecs which are in use to solve this problem. JPEG being the most commonly used codec for lossy image compression, many other codecs have gained popularity such as JPEG2000, Better Portable Graphics (BGP) and WebP. Use of Deep Neural Networks for the purpose of lossy compression of full resolution images has attracted a lot of attention in the recent years.

Deep neural network architectures which are commonly used for this purpose are autoencoders and recurrent neural networks (RNN). The key challenges in this field can be represented as 2 problems. One being the trade-off between entropy rate and reconstruction distortion of the latent representation of the image, and the other being the quantization of the image.

Work done by Mentzer et al. [2] tries to solve the challenge by addressing the first problem and proposing a new technique to control the entropy rate and reconstruction distortion trade-off for a new autoencoder for image compression. They propose an idea to use a context model i.e. a 3D-CNN to directly model the entropy of the latent representation of the image. It aims to learn the conditional probabilities represented by the latent distribution of the autoencoders. During training, the context model is continuously updated so that it can learn the dependencies between the symbols in the latent representations.

While work done by Li et al. [9] proposes a convolutional neural network-based image compression system and tries to exploit the fact that local information content in an image is spatially variant. Their work proposes the adaptation of bit rate of different parts of the image as per their local content, which is allocated as per content weighted importance map. Then, the sum of importance map can thus serve as a continuous alternative of discrete entropy estimation to control compression rate.

Both the works show great improvement in image compression techniques but do not talk about the time taken

in compression and their application in the real world on low power devices. Both the techniques prove to be

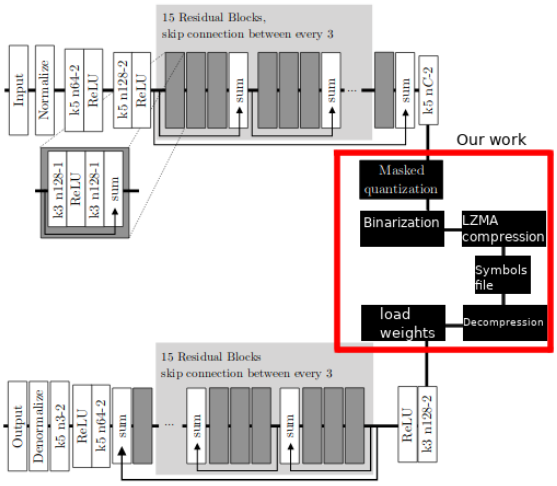

Figure 1: Image compression network by Mentzer et al [2] with our work highlighted

computationally expensive and irrelevant on devices with no graphical processing units (GPUs).

Techniques proposed by Johnston et al. [8] talks about improving the spatial diffusion by modifying the recurrent architecture and use of spatially adaptive bit allocation algorithm to encode visually complex regions in the image using limited number of bits, more efficiently.

Interesting work done by Prakash et al. [10] takes an out-of-the-box approach to the problem and tries to improve upon image compression ratios by creating a convolutional neural network which is specifically trained for the task of semantic image understanding which could be then used for higher visual quality in lossy compression.

2.3. Network compression

Application of Deep Neural Networks has helped us solve an immense number of problems across various fields of science and technology. In the recent years, deep neural networks have beaten the state-of-the-art and best human players in games such as Alpha-Go and chess. In the field of computer vision, neural networks have beaten human beings over accuracy in classification and recognition challenges.

Although these neural networks are very powerful and successful in the tasks they are trained for, the large size they occupy on hard drive and the immense computation power they demand make them less lucrative to be used on low power, computationally constrained devices such as smart phones. Networks like AlexNet and take up to 200MB of space on disc and VGG-16 can be as big as 500MB which makes them unsuitable to be used over small devices. Apart from taking more space and resources, their processing also demands lot of power which can drain the battery of portable devices.

Thus, in the recent years, research on compressing neural networks has got a lot of attention from researchers around the world. Work done by Han et al. [11] proposes a technique to prune the network by removing redundant connections and keeping only the most informative ones. It then, quantizes the weights so that multiple connections share the same weights so that only the effective weights and the indices are needed to be stored. Upon that, it proposes the application of Huffman coding to take the advantage of the biased distribution of the effective weights. In this approach, pruning and trained quantization are able to compress the network without interfering each other.

This process reduces the execution time taken by the neural network, improves its energy efficiency and reduces its storage space requirement of the neural network. Thus, this techniques improves the application ability of any neural network and makes it suitable for execution on computationally and power constrained devices.

3. Our Approach

To achieve efficient lossy image compression with our network we first implemented the autoencoder based design by Mentzer et al [2], a diagram of which can be found in Figure 1. The code for this network is publicly available for tensorflow and features a trainable autoencoder based network trained on bpp, ms-ssim, and psnr which are explained in section 5 of this paper. The authors of the network also provide pre-trained weights which we made use of in our evaluation. Using this network we are able to reduce the size of a directory of PNG images significantly.

Mentzer et al. appear more concerned with proving the viability of the network than actually putting it to use. By default the bpp of the network is estimated from the entropy of the network's latent space instead of by encoding it. Their optional implementation for arithmetic encoding however is very inefficient, taking up to 5 minutes for both encoding and decoding per image. As this is rather impractical for the task of image compression we aim to reduce this encoding time with a more efficient algorithm.

In order to achieve similar compression rates to Mentzer at a much higher speed we have utilized our own method of storing the latent space symbols. First, the latent space is binarized using Pickle, a Python library capable of binarizing arbitrary data structures. Once the latent space tensor is converted to a bitstream we use LZMA encoding to compress the stored file. When the file is needed again it can be decompressed, loaded back into Python and run through the decoder network.

The Lempel–Ziv–Markov chain algorithm (LZMA) is an algorithm used for lossless compression of data. It uses a dictionary compression algorithm whose output is encoded

with a range encoder. It does so by making a probability prediction for each bit. A dictionary compressor then converts the data into phrases and symbols which are then encoded one bit at a time using the range encoder. Since many encodings are possible, a dynamic programming algorithm is used to find the optimal encoding scheme given certain constraints. LZMA has better compression rates compared to Huffman and Shannon as it uses context specific to the bitfields in representing a literal or phrase. This avoids the mixing of unrelated bits together in the same context which is a possibility in the techniques mentioned latter which use a generic byte-based model.

In LZMA compression, streams of compressed bits are encoded using an adaptive binary range encoder. The streams are then divided into packets which represent either a single byte or an LZ77 sequence with its length encoded with it. Packets are then divided into parts which are then modeled with independent contexts so that the probability predictions of the bits are correlated with their values in the previous packets of the same type. [12]

With the problem of image compression taken care of our goal was to then compress the autoencoder network to increase its mobile compatibility. We were unfortunately unable to accomplish this as we found too many roadblocks to implementing optimizations during and after training.

4. Experiments

Since we strive to compare not only compression quality of our proposed techniques but also the time taken in the end to end process of compression and decompression, we will be applying our proposed method on 2 datasets. We will compress and decompress folders containing images from 2 datasets separately, using our proposed method and then do the same with the original image compression algorithm and other commercially viable data compression software and application such as 7-Zip and TAR. We would then compare all these techniques with each other on the basis of the time taken to compress and decompress the respective folders of the 2 datasets, along with the compression rate and compression quality achieved in the process.

4.1. Image compression losses

The end goal of the image compression network is to achieve a low number of bits per pixel (bpp) while retaining a high level of accuracy. Multiple accuracy measures are used, however the most common metric is Multi-scale structural similarity (MS-SSIM) which is known for being a metric that aligns well with human perception. MS-SSIM aside from being a good metric can also form our loss function so that we are optimizing for images that appear accurate to the human eye. It may be worth comparing MS-SSIM as a loss function to L2 loss to show the difference in overall quality of the reconstructed image. Adversarial loss has been explored in other papers such as [4, 5] but is out of scope of this paper.

4.2. Network compression metrics

The goal of network compression is to reduce the overall size of the network, often expressed as both a compression rate (higher is better) and a final size (lower is better). Pruning and quantization can be adjusted to explore the change in accuracy of the network at different levels of compression.

4.3. Data sets

While Kodak will be used to evaluate our results it is too small of a dataset to use for training. For this reason it may be worth experimenting on different training sets. For evaluating our compression at different scales we will train it using various image sizes from the ImageNet dataset and evaluate final performance and training time with various training sets.

Figure 2. Sample images of the Kodak data set.

5. Evaluation

Since our network is going to involve two different key philosophies for mobile image compression we will need to evaluate our networks performance on multiple fronts: achieved bitrate with respect to accuracy, the total compression rate and speedup of the network achieved, and the difference in bitrate and accuracy after applying our encoding scheme

5.1. Image Compression Losses

The end goal of the image compression network is to achieve a low number of bits per pixel (bpp) while retaining a high level of accuracy. Multiple accuracy measures are used; however the most common metric is Multi-scale structural similarity (MS-SSIM) which is known for being a metric that aligns well with human perception. MS-SSIM aside from being a good metric can also form our loss function so that we are optimizing for images that appear accurate to the human eye. It may be worth comparing MS-SSIM as a loss function to L2 loss to show the difference in overall quality of the reconstructed image. Adversarial loss has been explored in other papers such as [4, 5] but is out of scope of this paper.

5.2. Network Speed Metrics

Since the goal of the project is also to make the time taken by the network of image compression network shorter, we would also be calculating and monitoring the time taken for the network to execute in an attempt to quantify the speed of our network compared to other networks and compression techniques.

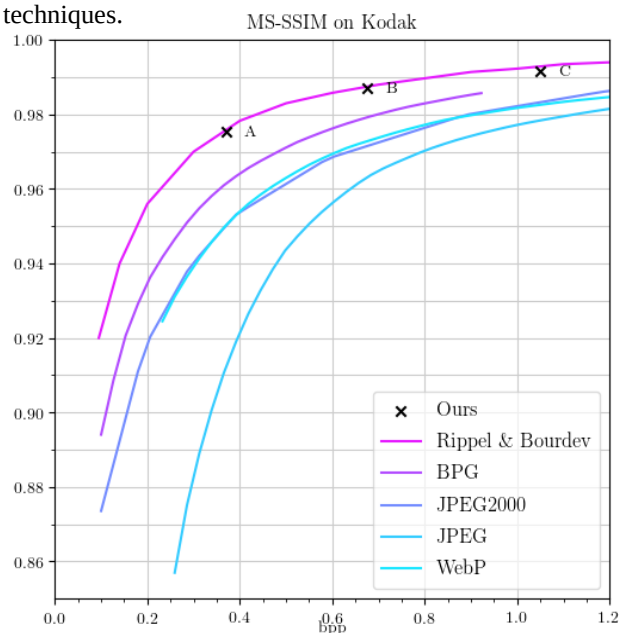

Figure 3. MS-SSIM wrt bpp at various points during training

5.3. Data Sets

We will be using 2 datasets to evaluate our proposed network compared to the original network and other image compressing tools.

Kodak dataset is the most ubiquitous dataset used to compare models and techniques in the field of Image compression. The dataset was released by the Eastman Kodak Company for unrestricted usage. The Dataset contains 24 lossless, true color (24 bits per pixel, aka "full color") images in PNG format.

Another dataset being used for evaluation and comparison of models and software in the field of image compression is the Validation Dataset P form CLIC i.e. Challenge on Learned Image Compression. CLIC is a competition sponsored by Google and conducted by CVPR which aims to boost the research in the field of image and network compression. Thus, we believe that the datasets provided for this competition will soon become industry standards in this field of research as they have an extensive dataset from 2 different kinds of sources i.e. Professional Photographers and from Mobile photography. These datasets are divided into 2 parts namely "Training" and "Validation". The first competition took place in 2018. Here, for our purpose we are using a validation dataset from P (a.k.a. professionals) as it has the smallest size and thus is easier to process on our limited processing power. The dataset consists of 41 uncompressed "true color" images

Figure 3. Plot of structural similarity wrt.bit per pixel rate. The three saved training checkpoints are marked A, B, and C.

6. Results

We compare 7-Zip, TAR, Mentzer et al., and our approach on 3 scales, namely total time taken in compression and decompression, image quality after compression, and image compression rate.

Mentzer et al. produces three checkpoints from its training period, it shows the effect of hyper-parameter tuning on the trade-off between MS-SSIM and BPP in the approach proposed in the research. These checkpoints are marked as A, B and C in Figure 3. Where:

- Point **A** in the plot (on Kodak: bpp: 0.370, MS-SSIM: 0.975)
- Point **B** in the plot (on Kodak: bpp: 0.677, MS-SSIM: 0.987)
- Point **C** in the plot (on Kodak: bpp: 1.051, MS-SSIM: 0.992)

We apply our techniques over all 3 of the checkpoints and compare them with the best of the original 3 checkpoints. It is worth noting that in order to better compare our results with 7z and tar and how they are typically applied we evaluate our results on an entire directory.

Technique	Avg. BPP per Image	Avg. MS-SSIM per Image	Avg. PSNR per Image	Total Time taken to Compress and Decompress
Mentzer et al.	0.370	0.992	27 – 30 dB	~ 14,000 sec
LZMA – Mentzer C	1.376	0.991	30.14 dB	94.54 sec
LZMA – Mentzer B	0.965	0.987	28.77 dB	78.93 sec
LZMA – Mentzer A	0.583	0.975	27.17 dB	72.13 sec

Table 1. Comparison of time taken and BPP with respect to Image quality after compression, with state-of-the-art techniques. (Kodak)

Technique	Avg. BPP per Image	Avg. MS-SSIM per Image	Avg. PSNR per Image	Total Time taken to Compress and Decompress
Mentzer et al.	0.370	0.992	27 – 30 dB	> 15,000 sec
LZMA – Mentzer C	1.285	0.991	31.26 dB	871.06 sec
LZMA – Mentzer B	0.833	0.987	30.02 dB	794.90 sec
LZMA – Mentzer A	0.467	0.978	28.58 dB	805.23 sec

Table 2. Comparison of time taken and BPP with respect to Image quality after compression, with state-of-the-art techniques. (CLIC - Professional)

Technique	Compressed Size	Compression Rate	Total Time taken to Compress and Decompress
7-Zip	15.4 MB	1	1.907 sec
TAR	15.4 MB	1	0.617 sec
LZMA – Mentzer C	1.6 MB	9.625	94.54 sec
LZMA – Mentzer B	1.1 MB	14	78.93 sec
LZMA – Mentzer A	0.688 MB	22.383	72.13 sec
Mentzer et al.	0.436 MB	35.32	~ 14,000 sec

Table 3. Comparison of compression rate with respect to total time taken, with commercially viable image compression software and applications (Kodak)

Technique	Compressed Size	Compression Rate	Total Time taken to Compress and Decompress
7-Zip	134.22 MB	1.005	12.525 sec
TAR	134.7 MB	1.001	5.074 sec
LZMA – Mentzer C	14.6 MB	9.24	871.06 sec
LZMA – Mentzer B	9.4 MB	14.58	794.90 sec
LZMA – Mentzer A	5.2 MB	25.96	805.23 sec
Mentzer et al.	3.5 MB	37.92	> 15,000 sec

Table 4. Comparison of compression rate with respect to total time taken, with commercially viable image compression software and applications (CLIC - Professional)

Results shown in Table 1 explains the comparison of our proposed technique applied on the 3 checkpoints given in Mentzer et al. with the best of Mentzer et al. results. It is evident that the proposed technique is faster by multiple folds and can produce results comparable to the original network proposed in Mentzer et al. especially in the application of LZMA-Mentzer C where a little sacrifice in

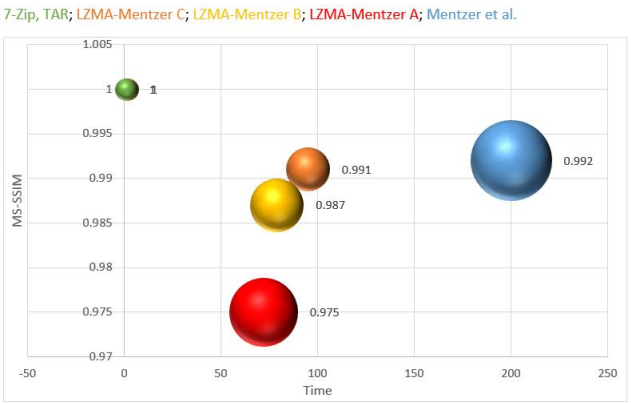

Figure 4. Bubble Graph showing compression Ratio with respect to MS-SSIM and Time for Kodak Dataset

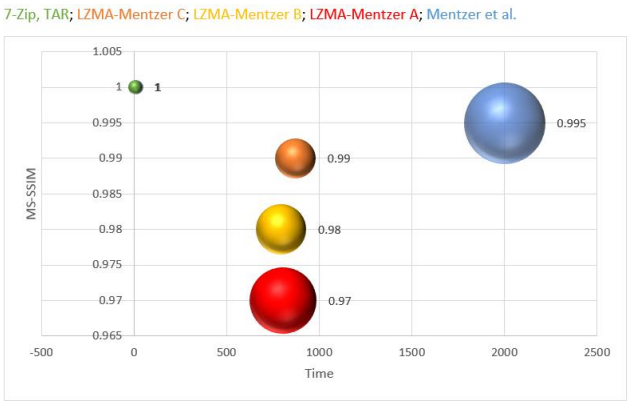

Figure 5. Bubble Graph showing compression Ratio with respect to MS-SSIM and Time for CLIC Dataset

MS-SSIM score can produce a comparable BPP at an astronomically faster speed. The results are further validated with experiments on the CLIC dataset which show a similar trend as given in Table 2.

The proposed method is then further compared with commercially used image compression software and applications in table 3 and 4. Our proposed method is compared with 7-Zip and TAR on the basis of compression rate and the total time taken to compress and decompress. Although the time taken by our proposed method is many folds more than the commercially used software but it compensates it by getting an equivalent feat in compression rates.

Figure 4. shows the bubble graph comparing all the discussed techniques on the basis of their execution time, after compressed image quality and compression rate for the Kodak Dataset while Figure 5. shows the same for CLIC.

7. Conclusion

Judging by the results we got from our experimentation, it is evident that the proposed technique is many times faster than the original technique with very little loss in compression quality and compression ratio compared to the original technique. The speeds are barely comparable to the techniques which are commercially available, but the compression ratios achieved are good enough to level the field.

Thus, we can conclude that the proposed method strikes a balance between compression quality, compression ratio and speed which can be deemed viable for practical uses. The proposed method competes with advancements in image compression techniques in the research academia and the fast execution times needed for commercial success and so, has achieved the aim of the research project.

8. References

[1] T. Shaham and T. Michaeli, "Deformation Aware Image Compression", *arXiv.org*, 2019. [Online]. Available: https://arxiv.org/abs/1804.04593v1. [Accessed: 11- Apr- 2019].

[2] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte and L. Van Gool, "Conditional Probability Models for Deep Image Compression", *arXiv.org*, 2019. [Online]. Available: https://arxiv.org/abs/1801.04260. [Accessed: 11- Apr- 2019].

[3] G. Toderici et al., "Full Resolution Image Compression with Recurrent Neural Networks", *arXiv.org*, 2019. [Online]. Available: https://arxiv.org/abs/1608.05148. [Accessed: 11- Apr- 2019].

[4] S. Santurkar, D. Budden and N. Shavit, "Generative Compression", *arXiv.org*, 2019. [Online]. Available: https://arxiv.org/abs/1703.01467. [Accessed: 11- Apr- 2019].

[5] E. Agustsson, M. Tschannen, F. Mentzer, R. Timofte and L. Van Gool, "Generative Adversarial Networks for Extreme Learned Image Compression", *arXiv.org*, 2019. [Online]. Available: https://arxiv.org/abs/1804.02958. [Accessed: 11- Apr- 2019].

[6] L. Theis, W. Shi, A. Cunningham and F. Huszár, "Lossy Image Compression with Compressive Autoencoders", *arXiv.org*, 2019. [Online]. Available: https://arxiv.org/abs/1703.00395. [Accessed: 11- Apr- 2019].

[7] O. Rippel and L. Bourdev, "Real-Time Adaptive Image Compression", *arXiv.org*, 2019. [Online]. Available: https://arxiv.org/abs/1705.05823. [Accessed: 12- Apr- 2019].

[8] N. Johnston et al., "Improved Lossy Image Compression with Priming and Spatially Adaptive Bit Rates for Recurrent Networks", *arXiv.org*, 2019. [Online]. Available: https://arxiv.org/abs/1703.10114. [Accessed: 12- Apr- 2019].

[9] M. Li, W. Zuo, S. Gu, D. Zhao, and D. Zhang, "Learning Convolutional Networks for Content-Weighted ion," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[10] A. Prakash, N. Moran, S. Garber, A. Dilillo, and J. Storer, "Semantic Perceptual Image Compression Using Deep Convolution Networks," *2017 Data Compression Conference (DCC)*, 2017.

[11] Mao, et al. "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding." *[Astro-Ph/0005112] A Determination of the Hubble Constant from Cepheid Distances and a Model of the Local Peculiar Velocity Field*, American Physical Society, 15 Feb. 2016, arxiv.org/abs/1510.00149.

[12] M. Mahoney, "Data Compression Explained", Mattmahoney.net, 2019. [Online]. Available: http://mattmahoney.net/dc/dce.html#Section_52. [Accessed: 26- Apr- 2019].