

Image Compression Using Neural Networks

CS 297 Report

Presented to

Dr. Chris Pollett

Department of Computer Science

San Jose State University

In Partial Fulfillment

Of the Requirements for the Class

CS 297

By Kunal Deshmukh

December 2018

ABSTRACT

Image compression is one of the most studied problem in computer science. Lossy image compression is being used to compress image size for transmission of images over the internet as well as their efficient storage. Convolutional Neural Networks have been used successfully in computer vision and image processing tasks but it is seldom used in tasks such as image compression. In this project, we will explore how image compression can be achieved using Deep Neural Networks which provides equivalent or better performance than traditional Image Encoding and data compression algorithms. We will show how Convolutional Neural Networks can be used along with Generative Adversarial Networks to improve image quality.

Keywords – Image Compression, Lossy image compression, Convolutional Neural Networks, Neural Networks, Generative Adversarial Networks.

TABLE OF CONTENTS

ABSTRACT	2
TABLE OF CONTENTS.....	3
I. INTRODUCTION	4
II. DELIVERABLE 1: IMPLEMENT END TO END IMAGE COMPRESSION ARCHITECTURE IN PYTORCH	7
III. DELIVERABLE 2: IMPLEMENT SINGLE IMAGE SUPER RESOLUTION USING GAN	9
IV. DELIVERABLE 3: MODIFY DELIVERABLE - 1 TO RUN ON IMAGENET DATASET. FIND SSIM, PSNR MATRICES.	11
V. DELIVERABLE 4: IMPLEMENT AND RUN FULL RESOLUTION IMAGE COMPRESSION USING RECURRENT NEURAL NETWORK	13
VI. CONCLUSION	15
REFERENCES.....	16

I. INTRODUCTION

More and more images are being captured in the world today, with the advent of Internet Of Things (IOT) and automation, even greater number of synthetic images are being generated by the machines. These tools not-only facilitate image capture, but also contribute to higher resolution images. These images occupy more space and require greater bandwidth for transfer as compared to a low-resolution image. Storage and transfer of such images is a challenge before industry. In this project, I aim to find improved way to compress an image using neural networks.

Even though image compression is an acute problem at present, it is in no-way a new problem. Study of image compression has led to many groundbreaking discoveries in data compression field so far such as Run-length Encoding and Discrete Cosine Transform (DCT). With the advent of neural networks, new neural net-based frameworks are being used for image compression [1].

In this semester, we read several papers to learn about previous attempts at neural network-based image compression. In the paper on end-to-end image compression, Feng et al. [2] propose Fully Convolutional Neural Network (FCNN) based Autoencoder. To do image compression, two separate networks are trained together for image compression and image decompression. Residual Learning is used in decompression network for faster learning process. Feng et al. claims framework delivers twice the image compression over JPEG images. In another approach employed by a team from twitter [3], Generative Adversarial Networks (GANs) are used to enhance quality of image. Original image can be simply downgraded by using Image library of python. The hope is this loss in image quality can be regained using GANs. Owing to the size of

network and number of trainable parameters, GANs are difficult to train. However, they are enormously useful in “retrieving” lost quality of the image.

Toderici et al. [4] attempt to use Recurrent Neural Networks (RNN) for image compression. This approach is suitable for portable devices where bandwidth may prove a bottleneck for graphic intensive websites. In this approach quality of images improve when more bits of information sent to it. This architecture is proven to provide better visual quality than JPEG, JPEG2000 and WebP formats while size of images is reduced by as much as 10%.

Convolutional Neural Networks (CNN) are a major component of all the architectures described above. CNNs are efficient in various Computer Vision tasks such as Image Recognition, Segmentation as well as compression. Various studies have proposed the way to improve efficiency of neural networks. Dat Thanh Tran proposed [9] the use of multilinear filters for convolution. This architecture uses several times less memory than CNN while outperforming CNNs. Landola et al. introduced SqueezeNet [10] architecture which uses significantly fewer parameters while achieving Alexnet level accuracy.

In this project, we aim to use these advances made in deep neural network to find image compression framework which could give best possible performance in image compression while having minimal overload. The organization of this report is divided into four deliverables that were accomplished in CS297. Each deliverable is designed to explore these avenues in detail. Deliverable 1 explores how fully convolutional variational autoencoders can be used for image compression and retrieval. In this deliverable, fully convolutional autoencoder is used. Deliverable 2 attempts to use GAN to obtain higher resolution image from lower resolution counterpart. This deliverable follows a paper on single image super-resolution using GAN.

Deliverable 3 examines results of architecture developed in Deliverable 1 on Imagenet pets' dataset with images resized to 256 x 256 size. This deliverable also involves a development python code to calculate SSIM, PSNR and MSE error between two images. In Deliverable 4, a paper on full resolution image compression using recurrent neural networks is implemented.

II. DELIVERABLE 1: IMPLEMENT END TO END IMAGE COMPRESSION ARCHITECTURE IN PYTORCH

The Objective of this deliverable was to implement an end-to-end image compression framework as prescribed by Tao et al. [3]. This framework consists of a compression layer called comCNN and reconstruction layer called recCNN. Both networks are fully convolutional neural networks.

This architecture is implemented in PyTorch framework. Google Colaboratory was used for its execution. The implementation was tested on 2 datasets - CIFAR and STL10. Both datasets were available in torchvision module.

Since the code is written in a jupyter notebook, it can be executed by selecting 'run all'.

This code uses GPU if available. The flag 'cuda' is set by using a function `is_available: torch.cuda.is_available()`. We used the Adam Optimizer. Since there is no high level function for interpolation in pytorch, a wrapper class is used which encapsulates interpolate function in `nn.functional` module. For ease of implementation, bilinear interpolation was used instead of bicubic interpolation.

The GPU memory is limited on google colab. Hence is it advised to not use batch size more than 16. Approximately 9GB GPU memory was used for the STL10 dataset.

The results obtained were as follows:

MSE: 3449.59, SSIM: 0.60



MSE: 1239.43, SSIM: 0.71



MSE: 3181.36, SSIM: 0.58



MSE: 2406.95, SSIM: 0.80



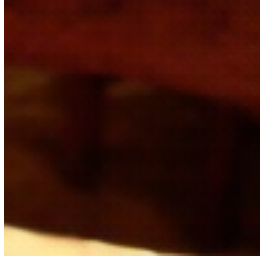





III. DELIVERABLE 2: IMPLEMENT SINGLE IMAGE SUPER RESOLUTION USING GAN

In this deliverable a single image super resolution algorithm proposed by C. Ledig [4] was implemented. We used the ready-made VGG network provided by torchvision module for discriminator network. As with any Generative Adversarial Network, the generator network of this framework is crucial for the task of super resolution.

Our architecture was implemented using pytorch framework and executed using google colab notebook as well as Google Cloud Platform. Images from the STL10 dataset were used for training as well as validation. A subset of images which were not used in training were used for inference purpose.

As the name suggests, class ‘generator’ and ‘discriminator’ are responsible for generator and discriminator respectively. ‘Residual_block’ and ‘unsampled_block’ classes are coded as per their definition in the research paper. Batch size was clapped to 16 due to limited resources.

The inference results obtained are as follows:

High resolution fake images	High resolution real images	Low resolution images
		
		



IV. DELIVERABLE 3: MODIFY DELIVERABLE - 1 TO RUN ON IMAGENET DATASET. FIND SSIM, PSNR MATRICES.

The objective of this deliverable was to train and test Deliverable 1 on higher resolution images and evaluate the results using matrices such as SSIM and PSNR. As a part of this deliverable, some experiments were performed on deliverable 1 and some changes were made in the framework. These experiments are as follows:

1. *Experiment setup:* Deliverable -1 was trained on few random images from ImageNet dataset. Since this is a huge dataset, only one subset of this dataset – pets’ images were considered for training as well as inference.

Observation: Since images were 200x200 in size as oppose to 32x32 images trained earlier, the program took significantly more time to execute each epoch.

Time required for the execution of 1 epoch with 50,000 32x32 images: 53 min

Time required for the execution of 1 epoch with ~9000 200x200 images: 2 hrs 12 min

Inference: Significant improvement in the performance of test and train images could be achieved if original images are divided in small parts and trained separately.

Since images were stored on Google drive, read speed might have caused the bottleneck.

2. *Experiment setup:* Average SSIM and PSNR error was calculated for inference images obtained from Deliverable -1 and Deliverable -4. When few changes were made in the architecture of Deliverable – 1. These changes include: Reduction in number of layers in RecCNN from 20 to 13., convolutional kernel size was changed from 3x3 to 2x2.

Design strategies used in SqueezeNet were adapted for this experiment.

Observation: The values are similar.

Inference: No significant change was observed with these changes.

As a part of this deliverable, a script to calculate image quality metrics (PSNR, SSIM etc.) was written as well. This script requires relative address of two images that needs to be compared and returns the quality scores as well as bit rates of each image as an output.

V. DELIVERABLE 4: IMPLEMENT AND RUN FULL RESOLUTION IMAGE COMPRESSION USING RECURRENT NEURAL NETWORK

The objective of this deliverable was to implement the architecture suggested in the paper [2]. This architecture provides variable compression rates. There are two kinds of architectures possible for reconstruction network — One shot and additive construction. This is a first architecture to outperform JPEG across most bitrates.

Even though this architecture uses Mean Square Error to find out loss between an image. Various image quality measurements such as Structural Similarity Matrix (SSIM) and Peak Signal to Noise Ratio (PSNR) was also evaluated.

Because of multiple invocations of recurrent neural networks layers GPU usage in google colab far exceeded the one required for this network to run. Hence, google cloud platform was used to run the compression network.

Since RNN and CNN are implemented together, according to the code as below:

```

hx, cx = hidden
gates = self.conv_ih(input) + self.conv_hh(hx)




ingate, forgetgate, cellgate, outgate = gates.chunk(4, 1)

ingate = F.sigmoid(ingate)
forgetgate = F.sigmoid(forgetgate)
cellgate = F.tanh(cellgate)
outgate = F.sigmoid(outgate)

cy = (forgetgate * cx) + (ingate * cellgate)
hy = outgate * F.tanh(cy)

```

This architecture does not take into account the human vision centric image accuracy matrices such as PSNR-HVS.

Original Image	Reconstructed image after second iteration	Reconstructed image after fourth iteration
		

VI. CONCLUSION

In this semester, I explored various image compression frameworks and existing file encoding formats. Implementation of various Neural network frameworks helped me understand to design and train non-standard neural network architectures.

I have written a code for three deep neural network frameworks used in image compression so far. This has led to development of large cache of tools e.g. code to read images folders / serialized dataset, a code snippet that can be used to obtain an image from tensor and save it in a desired format etc. These are fairly general-purpose codes which can be used to perform similar operations in CS298. E.g, I will be easily able to use serialized dataset provided using TorchVision module.

In CS298, I will use the concepts I learned in CS297, e.g, use of skip connections to speed up the training process, use of tanh followed by step function to obtain binary form of a latent representation etc. The neural networks that I implemented in CS 297 can be partially reusable in a new architecture. I will also use and compare various image quality matrices learned in this semester to design loss function.

REFERENCES

- [1] O. Rippel and L. Bourdev, "Real-Time Adaptive Image Compression," *The 34th Int. Conf on Mach. Learn.*, 2017. doi: arXiv:1705.05823v1.
- [2] G. Toderici et al., "Full Resolution Image Compression with Recurrent Neural Networks," *arXiv e-prints*, 2016. doi: arXiv:1608.05148.
- [3] W. Tao et al., "An End-to-End Compression Framework Based on Convolutional Neural Networks," *Data Compression Conf. (DCC)*, Snowbird, UT, 2017, pp. 463-463. doi: 10.1109/DCC.2017.54.
- [4] C. Ledig, et al., "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network.," *Conf. on Comp. Vision and Pattern Recog.* Vol. 2. No. 3, 2017. doi: arXiv:1609.04802.
- [5] J. Chung et al., "A recurrent latent variable model for sequential data," *In Advances in neural inform. process. syst.*, pp. 2980-2988. 2015.
- [6] D. Minnen et al., "Joint autoregressive and hierarchical priors for learned image compression". *Advances In Neural Inform. Process. Syst.*, 2018. doi: arXiv:1809.02736.
- [7] R. Pascanu et al., "How to construct deep recurrent neural networks", *In Proc. of the Second Int. Conf. on Learning Representations (ICLR 2014)*., 2014. doi:arXiv:1312.6026.
- [8] P. Cheng, "A New Single Image Super-Resolution Method Based on the Infinite Mixture Model", *Access IEEE*, vol. 5, pp. 2228-2240, 2017. doi: 10.1109.
- [9] Thanh Tran, Dat & Iosifidis, Alexandros & Gabbouj, Moncef. (2017). "Improving Efficiency in Convolutional Neural Network with Multilinear Filters. Neural Networks." *105. 10.1016/j.neunet.2018.05.017.*
- [10] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, et al. (2016). "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size." *arXiv:1602.07360.*