MAJOR PROJECT 2 - IMAGE COMPRESSION USING DEEP LEARNING

Jaypee Institute of Information Technology, Noida

Nishit Anand (9918103133), Shreya Agarwal (9918103146), Aditi Dixit (9918103155) Supervisor – Mr. Rupesh Kumar Koshariya

Problem Statement

In today's world, most of the data is in the form of images. We take so many photos with our smartphones. Different social media platforms like Instagram are driven by multimedia data. Images take a lot more storage than text files. All this data is stored either on our devices or on the servers of companies like Meta, Google, Twitter. Thus, reducing the space taken by them and compressing them would not only benefit MNCs by saving them storage space on their servers, but also us as our mobiles and laptops would be able to store more content. Thus, in today's day and age, image compression has become a topic which needs to be put light upon.

Proposed Solution

Our proposed solution is to use GANs (Generative Adversarial Networks) and CNNs (Convolutional Neural Networks) for learned image compression. First CNN are fed the images from which they learn the important features in the image. Then they pass on these extracted features to the GAN. In GAN, the Generator and Discriminator, work on these extracted features and create a new image from scratch which has only the important details and gets rid of less useful information. Thus, giving us a compressed image which looks perceptually the same as the original image, but is much smaller in size.

Our Solution: In Detail

1. Collecting the datasets: We collected the Kodak, CLIC2020 and DIV2K datasets and a subset of the ImageNet Dataset for training and testing our model.

2. Detect and Extract Features: We wrote a ML model made up of Convolutional layers and Maxpool2D layers to extract spatial features from the images

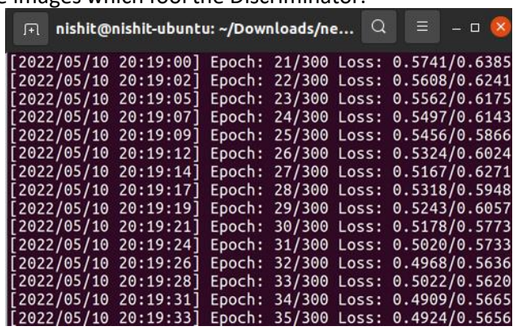
3. Generator learns and produces new images: These extracted features are then input into the GAN. The Generator starts learning from the features given by the CNN and tries to generate new example images. It tries to maximize the probability of the Discriminator making a mistake and thinking that the image produced by the it is from the training set instead.

4. Discriminator starts assessing the images produced by the Generator: Discriminator assesses the image produced by the Generator and tries to identify whether the image is real (i.e. from the training dataset) or fake (i.e. generated by the Generator). It returns its assessment to the Generator, in turn teaching the generator to generate images which fool the Discriminator.

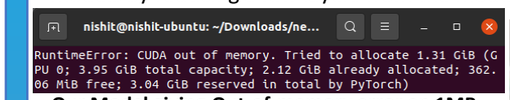
5. Training the model: We trained the model for 20,000 iterations for 36 hours. This was done in order to make the model more generalized and so that it is better able to learn the important portions in an image.

6. Output Compressed Images: After the Generator and Discriminator have been trained completely, then they produce images which are much smaller in size compared to original ones, but of almost the same quality visually.

7. Optimizing the model: Earlier our model could only compress images upto 500kB and upto 400x400 resolution. This was because it used a lot of GPU memory and our laptop had less GPU VRAM as compared to server GPUs. Because of this, it gave out of memory (OOM errors) sometimes. So we optimized our model and made it lightweight compared to other image compression models by reducing a few layers from the model.



Our Model training on our dataset



Our Model giving Out-of-memory error on 1MB image before optimization

Property of the Model	Our Model Before Optimization	Our Model After Optimization
Maximum Image size which could be compressed	500kB	1500kB (1.5 MB)
Maximum Image resolution which could be compressed	400x400 pixels (160,000 pixels)	1000x1000 pixels (1,000,000 pixels)

Table - Our model before and after optimization

Before Compression (1.1 MB)

After Compression (34.0 KB)

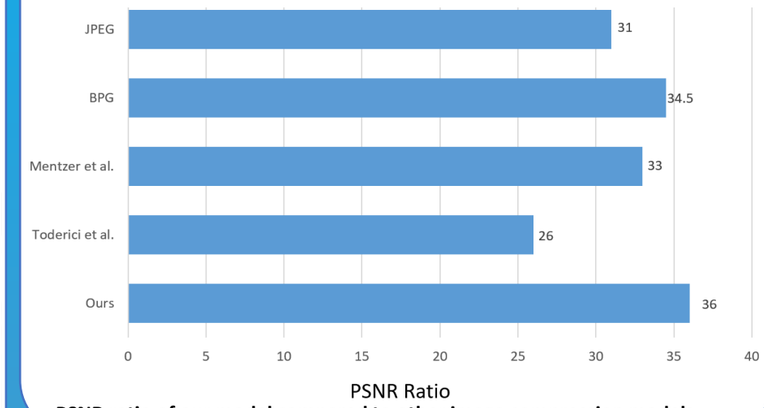
Results and Testing

We changed hyperparameters and fine tuned the model in order to decrease the output size of images and to increase the PSNR ratio of the output images.

We can see that our model compresses images by a lot and we were able to achieve a compression ratio of 11x smaller to even 42x smaller as compared to the original image.

Image Size Before Compression	Image Size after Compression	Compression Ratio (Original/Compressed)
1500kB	35.0kB	42.85
1000kB	24.7kB	40.48
500kB	17.8kB	28.09
100kB	8.7kB	11.49

Table - Comparison of Image size before and after compression by our model



PSNR ratio of our model compared to other image compression models

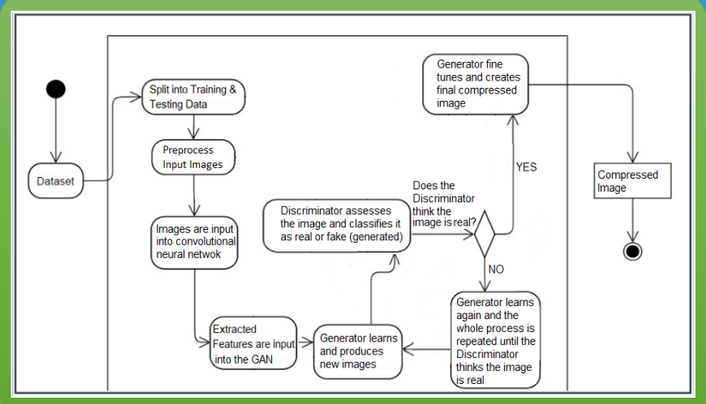
Conclusion

We have proposed GAN (Generative Adversarial Networks) based image compression which gives us a compressed image that looks perceptually the same as the original image, but is much smaller in size. Unlike existing methods, we have used Generative Adversarial Networks, which help reduce the size of output image by quite a bit while maintaining image quality.

In the end, our model became robust and was able to find which information in the image is important and needs to be included in the compressed image. In the future, we will attempt to compress images with even less resources and amount of GPU VRAM.

Before Compression (1.5 MB)

After Compression (35.0 KB)



Flowchart of our model

References-

[1] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell, "Full Resolution Image Compression with Recurrent Neural Networks",CVPR 2017

[2] Renjie Zou, Chunfeng Song, Zhaoxiang Zhang, "The Devil Is in the Details: Window-based Attention for Image Compression",CVPR 2022

[3] Khawar Islam, L. Minh Dang, Sujin Lee, Hyeonjoon Moon, "Image Compression with Recurrent Neural Network and Generalized Divisive Normalization", CVPRW 2021

[4] Myungsoo Song, Jinyoung Choi, Bohyung Han, "Variable-Rate Deep Image Compression through Spatially-Adaptive Feature Transform", ICCV 2021

[5] Yueqi Xie, Ka Leong Cheng, Qifeng Chen ,"Enhanced Invertible Encoding for Learned Image Compression", ACMMM 2021