IMAGE COMPRESSION USING DEEP LEARNING

Enrollment numbers: - 9918103133, 9918103146, 9918103155

Name of Students: - Nishit Anand, Shreya Agarwal, Aditi Dixit

Name of Supervisor: - Mr. Rupesh Kumar Koshariya

MAY - 2022

Submitted in partial fulfillment of the Degree of

Bachelor of Technology

in

Computer Science Engineering

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING & INFORMATION TECHNOLOGY

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

(I)

Table of contents

(II)

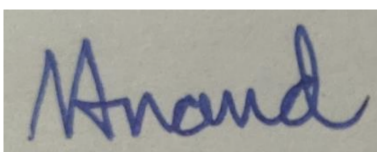
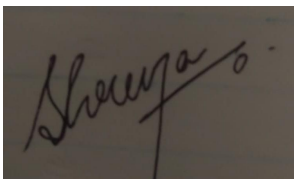
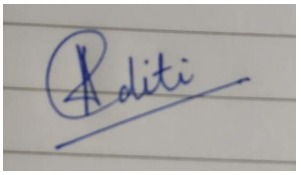
Declaration

We hereby declare that this submission is our own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: Noida

Date: 15-05-2022

Students-

Signature:			
Name:	Nishit Anand	Shreya Agarwal	Aditi Dixit
Enrollment No:	(9918103133)	(9918103146)	(9918103155)

(III)

CERTIFICATE

This is to certify that the work titled "**Image Compression using Deep Learning**" submitted by "**Nishit Anand, Shreya Agarwal , Aditi Dixit** " in partial fulfillment for the award of degree of Bachelors of Technology of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor

Name: Rupesh Kumar Koshariya

Designation: Assistant Professor (Grade -1)

Date: 15-05-2022

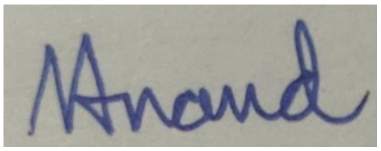
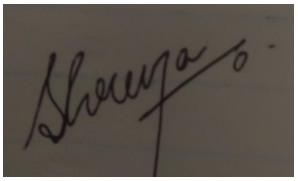
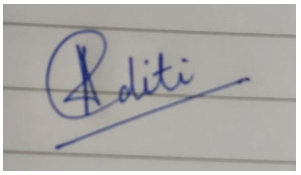
(IV)

ACKNOWLEDGEMENT

We deeply acknowledge and extend our thanks to our mentor - Mr. Rupesh Kumar Koshariya, to the Department Coordinator - Dr. Charu Gandhi, HOD CSE - Dr. Vikas Saxena, Director - Dr. Hari Om Gupta and to the Vice Chancellor - Dr. Yog Raj Sood, for their immense support, guidance and constant supervision as well as for providing necessary information regarding the project. We also are grateful for our teachers for providing us with this opportunity to present before them. We would also like to express our gratitude towards our parents and friends for their kind cooperation and encouragement which helped us in completion of this project.

Lastly we would like to extend our sincere thanks to everyone who helped us directly and indirectly in completion of our project.

Sincerely,

Signature:			
Name:	Nishit Anand	Shreya Agarwal	Aditi Dixit
Enrollment No:	(9918103133)	(9918103146)	(9918103155)

Date: 15-05-2022

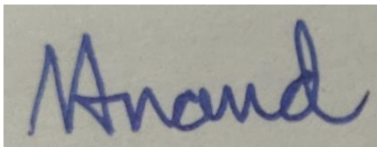
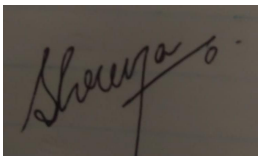
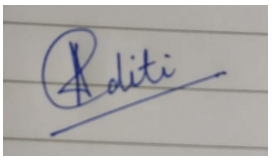
(V)

SUMMARY

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 would be very beneficial 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. Image compression deals with minimizing the number of bits required to represent an image. In this report, we have proposed our idea to create a framework for image compression. Unlike existing methods, we have used Generative Adversarial Networks, which help reduce the size of output image by quite a bit as well as help in maintaining image quality.

Signature of Students-

Signature:			
Name:	Nishit Anand	Shreya Agarwal	Aditi Dixit
Enrollment No:	(9918103133)	(9918103146)	(9918103155)

Date: 15-05-2022

Signature of Supervisor-

Name: Rupesh Kumar Koshariya

Date: 15-05-2022

(VI)

LIST OF FIGURES

(VII)

LIST OF TABLES

(VIII)

LIST OF SYMBOLS & ACRONYMS

Acronym	Full Form
GAN :	Generative Adversarial Networks
PSNR :	Peak Signal to Noise Ratio
BPG:	Better Portable Graphics
CNN:	Convolutional Neural Networks
TF:	Tensorflow
GAN:	Generative Adversarial Networks
CLIC2020:	Challenge on Learned Image Compression 2020
DIV2K:	DIVerse 2K Image Dataset
LTS:	Long term support
GPU:	Graphics processing unit
VRAM:	Video Random Access Memory
HEVC:	High Efficiency Video Coding
RNN:	Recurrent Neural Networks
SOTA:	State of the Art
FAIR:	Facebook Artificial Intelligence Research Group
OOM:	Out Of Memory Error

Chapter 1 : Introduction

1.1 General introduction

The increase in the number of cameras and the increase in the number of smartphones has led to a rapid growth in the number of image data. To store images efficiently, we use lossy image compression algorithms like JPEG. Instead of storing the raw RGB data, a JPEG file stores a lossy version of the image, with minimal visual changes as compared to the original image.

Various image compression algorithms have been proposed, including using state of the art video compression algorithms like HEVC (High Efficiency Video Coding) for single image compression.

But at the same time, deep learning based lossy image compression algorithms have also seen an increase in interest. In these algorithms, a neural network is optimized for the rate-distortion trade off. But these approaches degrade image quality by quite a bit.

We propose GAN (Generative Adversarial Networks) based image compression which does not degrade the image quality and gives results which are visually identical to the original image, while compressing the image a lot.

1.2 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. Image compression deals with minimizing the number of bits required to represent an image.

1.3 Significance of the problem

1. Image Compression refers to minimizing the number of bits required to represent an image, thus it reduces the amount of storage needed to save these pictures and also reduces the bandwidth required to transmit these pictures.

2. Less bandwidth requirement is important as wireless technology typically lags behind user trends which have been continuously requiring even higher bandwidths in the past few years and this is only set to increase with emerging technologies such as VR virtual reality.
3. Also better compression technology helps in improving accessibility in different developing areas of the world where the wireless infrastructure is much less performant and much less robust than in the developed countries.

1.4 Empirical Study

We studied the literature pertaining to the various image compression techniques by going through the existing research papers. We then tried reproducing the results of a few authors from the below mentioned papers. We analyzed different data sets used for image compression techniques. More details about the datasets have been mentioned further in the report.
We learnt how to combine CNNs (Convolutional Neural Networks) with GANs (Generative Adversarial Networks) to obtain a generative lossy image compression system which is able to perform well.
Also for deep learning we will be using NeuralNetworks and Convolutional Neural Networks. This requires a good amount of computational power and GPU power to do the required ML tasks quickly and efficiently.

1.5 Brief description of the solution approach

Our proposed solution is to use GANs (Generative Adversarial Networks) and CNNs (Convolutional Neural Networks) for learned image compression. First Convolutional Neural Networks are fed the images from which they learn the important features in the image. Then they pass on these extracted features to the Generative Adversarial Network. In Generative Adversarial Network, 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.

1.6 Comparison of existing approaches the problem faced

We studied the literature pertaining to learned image compression and read many research papers. In the past image compression has been done using Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) etc. We have used GANs (Generative Adversarial Network)

because GANs don’t break down with high-dimensional data. They work well with unsupervised data. Another advantage is that both networks, Generator and the Discriminator in GANs can be trained with only Backpropagation.

In Generative Adversarial Network, 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.

In contrast to previous work-

1. We obtain visually pleasing compressed images that are perceptually similar to the input
2. Our approach can be applied to high-resolution images well

Chapter 2: Literature Survey

2.1 Summary of papers studied

We studied and analyzed the following research papers as a part of a literature survey.

1. **"End-to-End Learned Block-Based Image Compression with Block-Level Masked Convolutions and Asymptotic Closed Loop Training"**

   by Fatih Kamisli

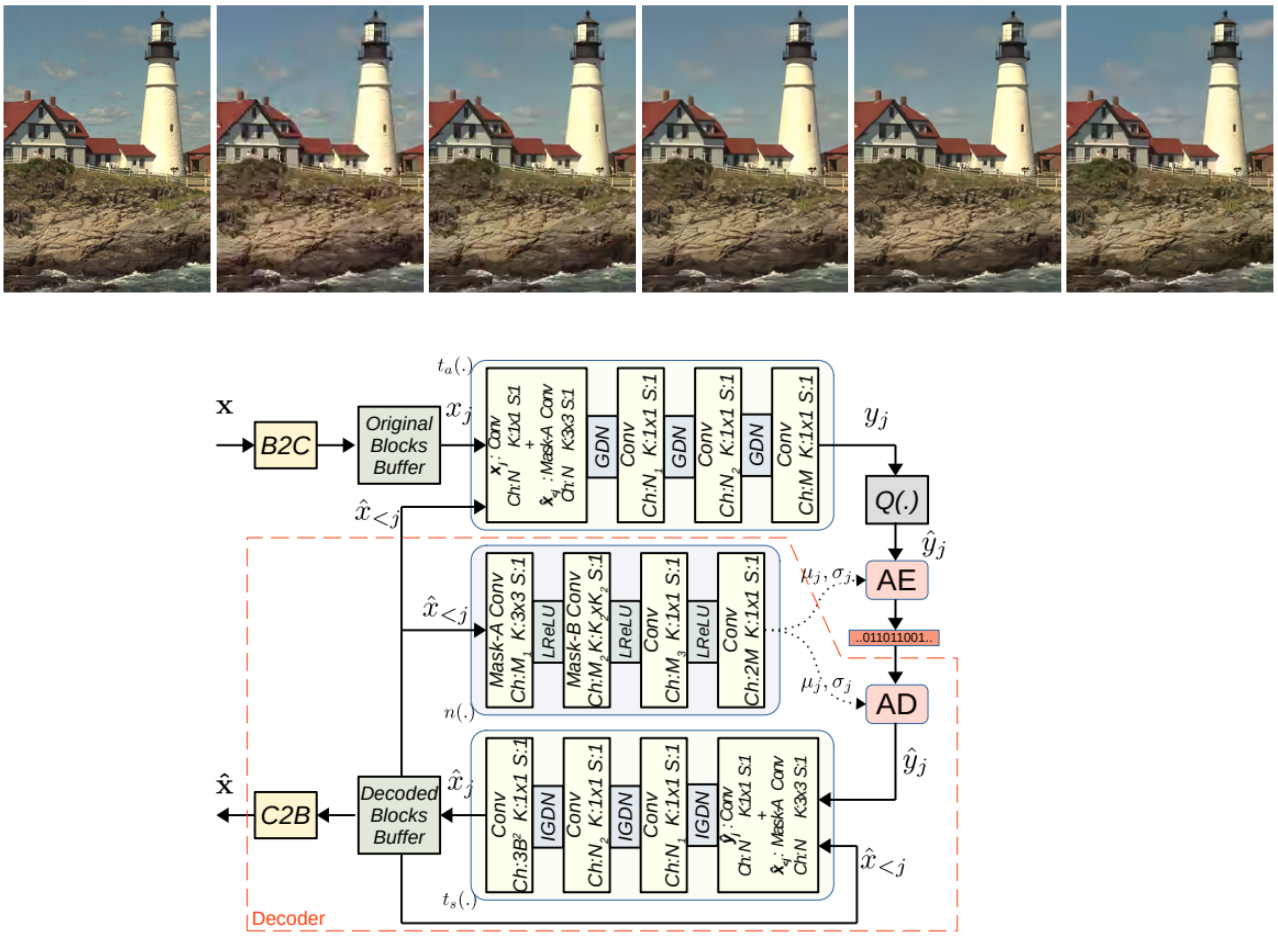


Figure 2.1 - End-to-End Learned Block-Based Image Compression with Block-Level Masked Convolutions and Asymptotic Closed Loop Training

<u>Summary:</u>

Image compression research has achieved state-of-the-art compression performance with auto-encoder based neural network architectures, in which the image is mapped into a latent representation via convolutional neural networks ("CNN"), quantized, and processed again with "CNN" to obtain the reconstructed image. "CNN" performs operations on complete

input pictures. Traditional state-of-the-art image and video compression methods, on the other hand, process pictures block by block for a variety of reasons. Work on learning image compression with block-based techniques has also recently emerged, which employ the auto-encoder architecture on large blocks of the input picture and include additional neural networks that provide intra/spatial prediction and deblocking/post-processing tasks. This research investigates an alternate learnt block-based picture compression method that does not rely on an explicit intra prediction neural network. This research investigates an alternate learnt block-based picture compression method that does not include either an explicit intra prediction neural network or an explicit deblocking neural network. A single auto-encoder neural network with block-level masked convolutions is utilized, with substantially smaller block sizes (8x8). Each block is processed utilizing reconstructed nearby left and upper blocks at both the encoder and decoder using block-level masked convolutions. As a result, during compression, the mutual information between adjacent blocks is used, and each block is rebuilt using nearby blocks, eliminating the requirement for explicit intra prediction and deblocking neural networks. Because the investigated system is a closed loop system, an asymptotic closed loop design is employed in conjunction with normal stochastic gradient descent-based training. The experimental results indicate competitive image compression performance.

2. **"Unified Multivariate Gaussian Mixture for Efficient Neural Image Compression"**
   By Xiaosu Zhu, Jingkuan Song, Lianli Gao, Feng Zheng, Heng Tao Shen

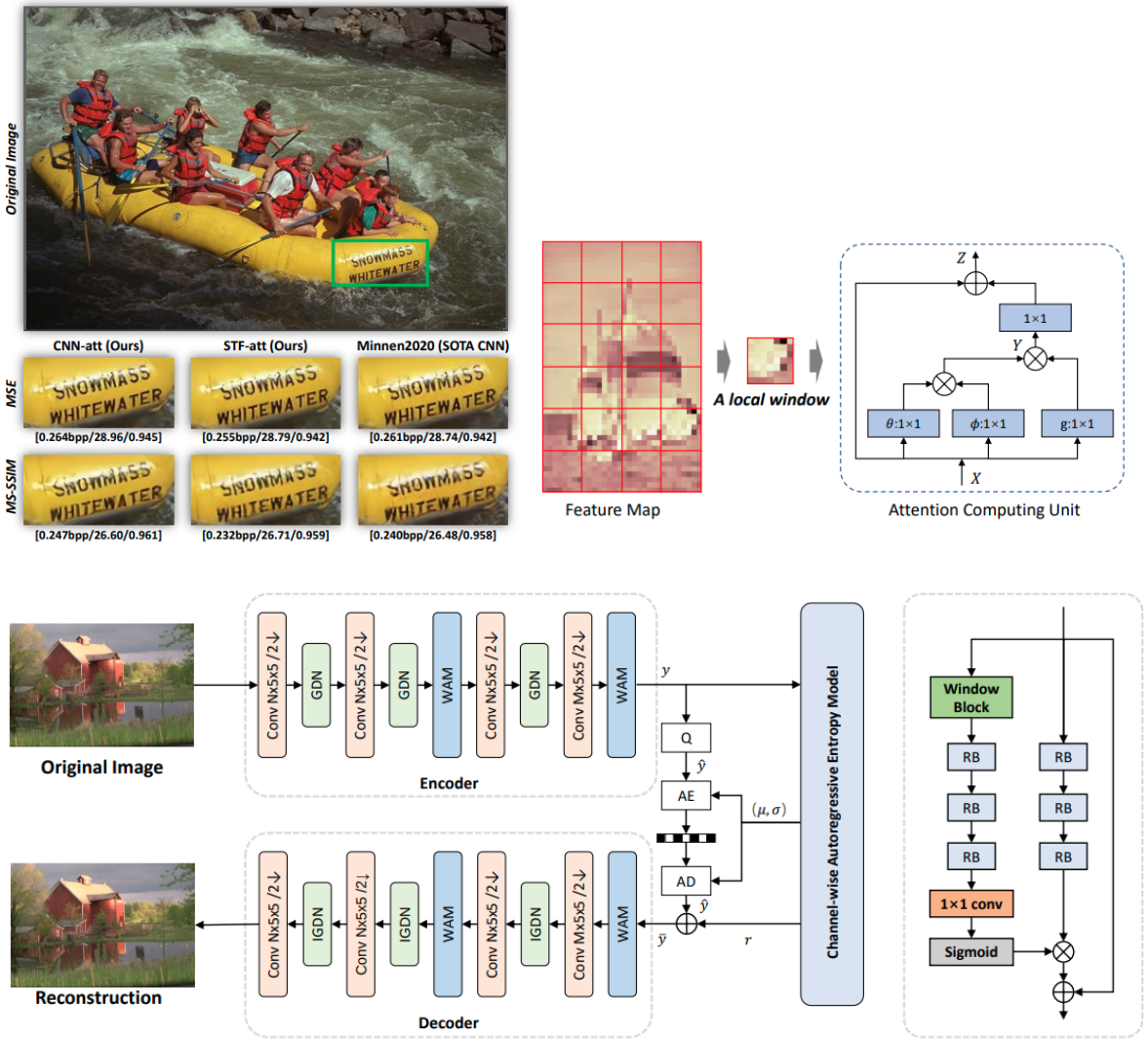
Figure 2.2 - Unified Multivariate Gaussian Mixture for Efficient Neural Image Compression

Summary:

The difficulty of modeling latent variables using priors and hyperpriors is critical in variational picture compression. Formally, if priors and hyperpriors properly characterize latent variables, the trade-off between rate and distortion is efficiently handled. Currently, only univariate priors are used, and each variable is processed independently. However, while examining latent variables in a vectorized viewpoint, this work discovers inter- and intra-correlations. These findings highlight visual redundancies that may be used to increase rate-distortion performance and parallel processing capability to accelerate compression. This motivates us to develop a new vectorized prior. A multivariate Gaussian mixture with estimated means and covariances is proposed. Then, without using context models, an unique probabilistic vector quantization is used to successfully estimate means, and leftover covariances are further induced to a unified mixture and solved by cascaded estimation.

3. **"The Devil Is in the Details: Window-based Attention for Image Compression"**

By Renjie Zou, Chunfeng Song, Zhaoxiang Zhang



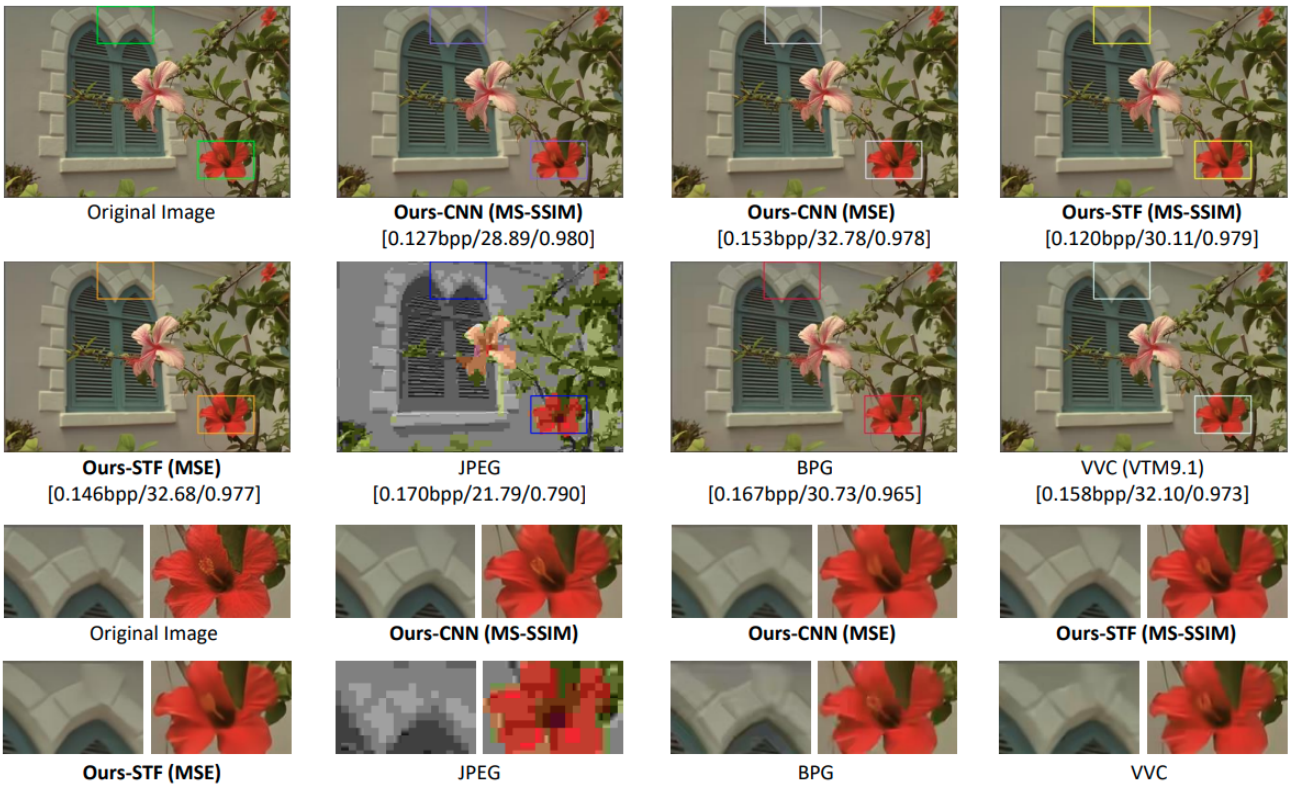


Figure 2.3 - The Devil Is in the Details: Window-based Attention for Image Compression

Summary:

Learned image compression approaches outperformed traditional image compression benchmarks of rate-distortion performance. Convolutional Neural Networks are the backbone of the majority of extant learnt image compression methods ("CNN "s). Despite significant advances, one major disadvantage of the "CNN"-based model is that its structure is not optimized for capturing local redundancy, particularly non-repetitive textures, which has a negative impact on reconstruction quality. As a result, how to fully utilize both global structure and local texture becomes the central issue for learning-based picture reduction. Inspired by recent advances in Vision Transformer (ViT) and Swin Transformer, this work discovered that combining the local-aware attention mechanism with global-related feature learning might achieve image compression expectations. The authors of this study first thoroughly investigated the impacts of several types of attention mechanisms for local feature learning, before introducing a more easy yet effective window-based local attention block. The suggested window-based attention model is relatively adaptable and might be used as a plug-and-play component to improve "CNN" and Transformer models. Furthermore, we present a new Symmetrical TransFormer (STF) framework with absolute transformer blocks in the down- and up-sampling encoders and decoders. Extensive experimental assessments have revealed that the suggested strategy is effective and outperforms current methods.

4. SC2: “Supervised Compression for Split Computing”

By Yoshitomo Matsubara, Ruihan Yang, Marco Levorato, Stephan Mandt

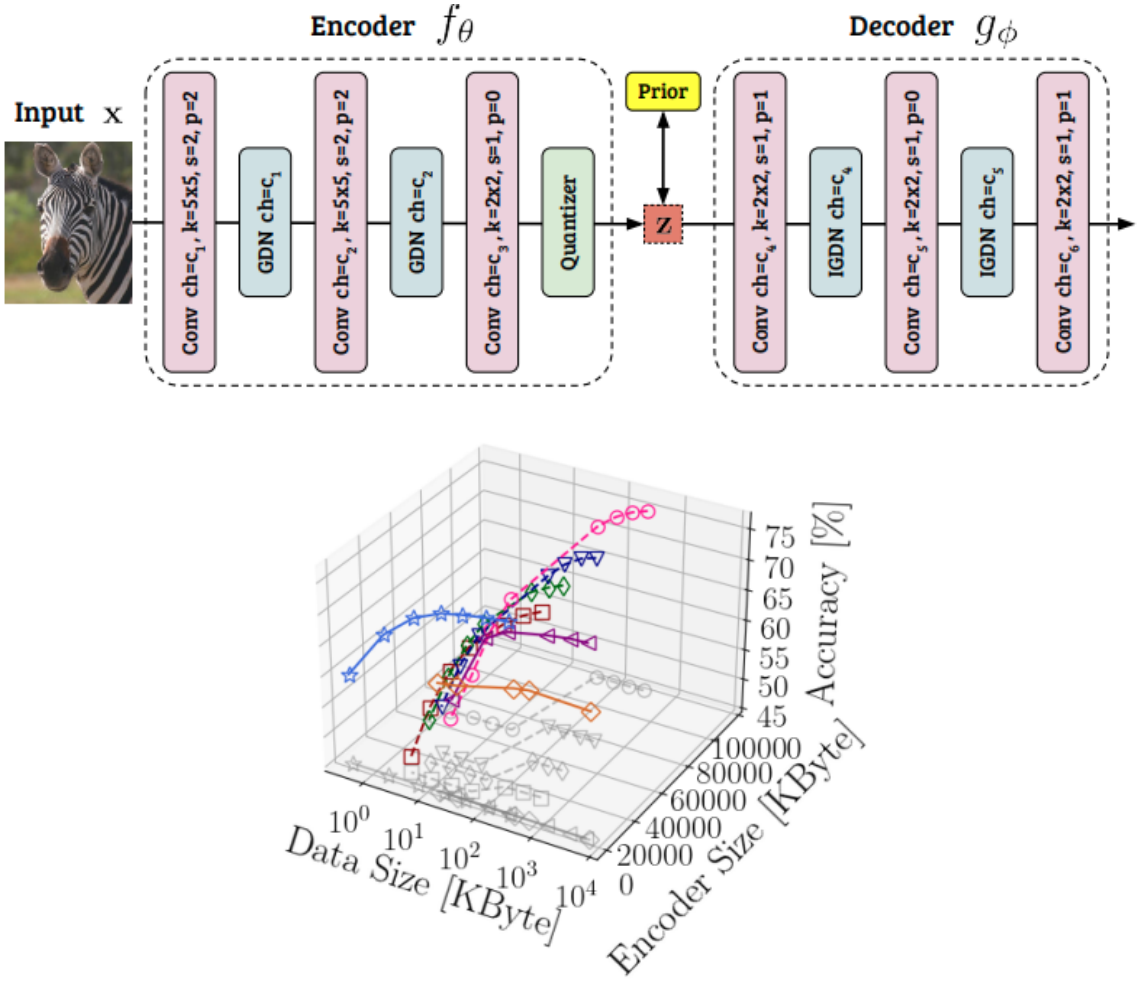


Figure 2.4 - SC2: Supervised Compression for Split Computing

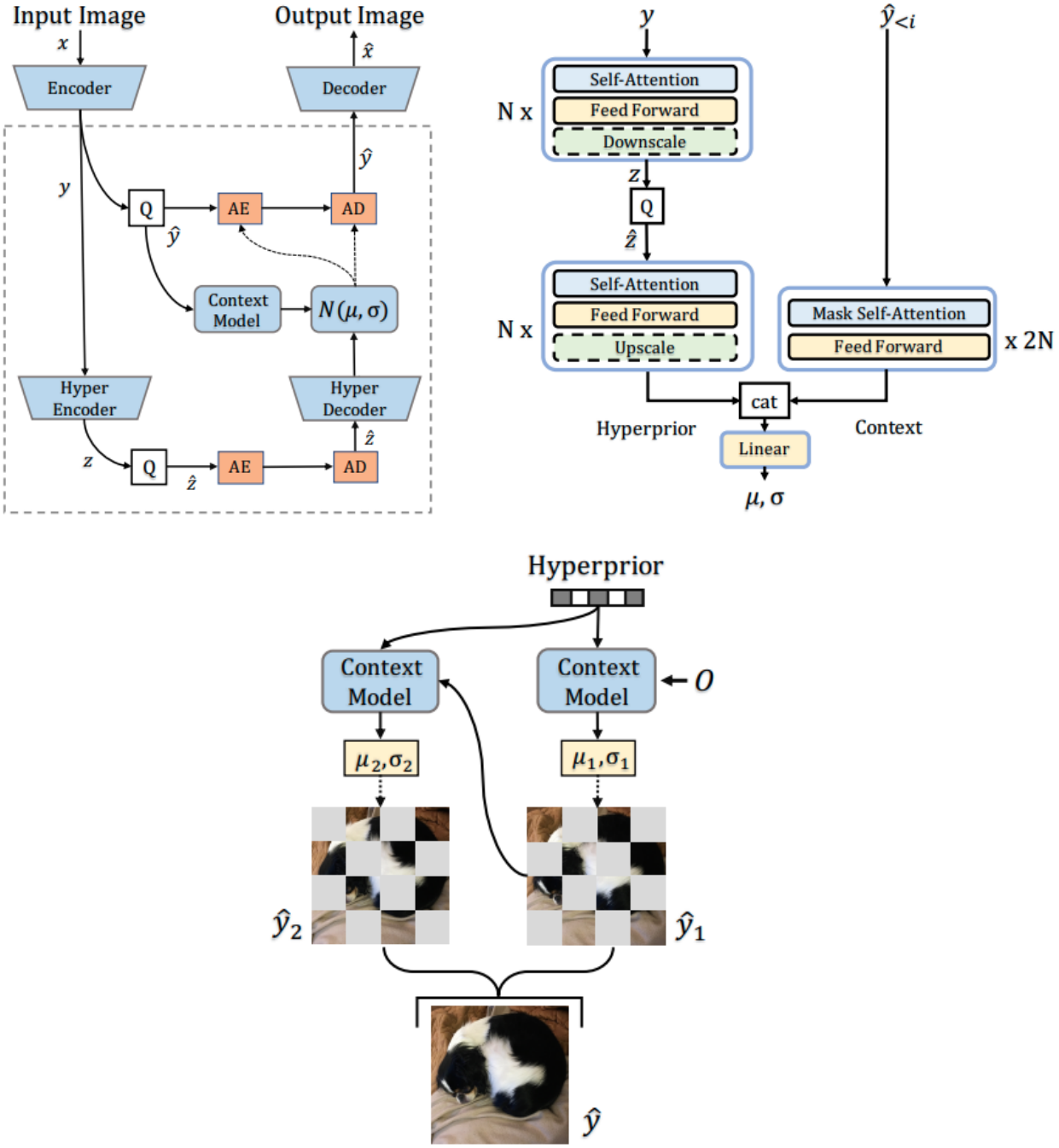
Summary:

Split computing spreads neural network execution (for example, for a classification job) between a mobile device and a more capable edge server. The supervised job may be completed entirely on the edge server while compressing and delivering the whole data set, and most techniques have barely outperformed this baseline. This study presents a new method for discretizing and entropy-coding intermediate feature activations in order to send them from the mobile device to the edge server as efficiently as possible. We demonstrate that an efficient splittable network design is the result of a three-way tradeoff between (a) limiting computation on the mobile device, (b) minimizing data size to be transferred, and (c) optimizing the model's prediction performance. Based on this tradeoff, this research offers an architecture and training of the positionable structure and entropy modeling in a data extraction framework. The authors demonstrate that their strategy improves supervised rate-distortion tradeoffs while keeping a significantly reduced encoder size in a large set of

experiments encompassing three computer vision algorithms, three samples, nine baselines, and more than 180 trained models. They also published sc2 bench, a Python tool that may be installed to promote and enable further research on monitored compression for split computing (SC2).

5. **"Entroformer: A Transformer-based Entropy Model for Learned Image Compression"**

By Yichen Qian, Ming Lin, Xiuyu Sun, Zhiyu Tan, Rong Jin



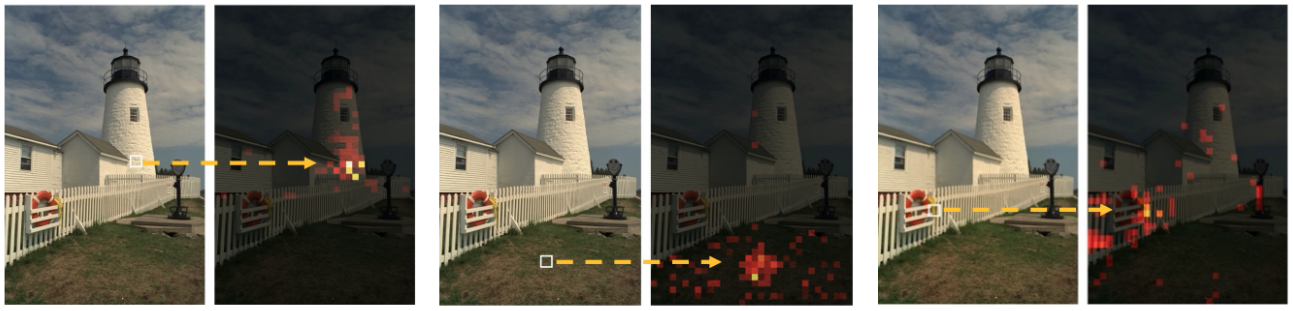


Figure 2.5 - Entroformer: A Transformer-based Entropy Model for Learned Image Compression

Summary:

The entropy framework, which forecasts the probabilistic model of the quantization latent space in the authentication and encryption modules, is a fundamental component of lossy deep image compression. Previous research relies on cnn models to create entropy models, which are poor in capturing global relationships. In this paper, the authors introduced Entroformer, an unique transformer-based entropy model for effectively and efficiently capturing long-range relationships in probability distribution estimation. The Entroformer, unlike vision changers in image classification, is highly tuned for picture compression, with top-k self-attention and diamond relative position coding. Meanwhile, they enhanced this structure with a concurrent two way context model to accelerate decoding. The trials reveal that the Entroformer provides cutting-edge picture compression performance while being time-efficient.

6. "Parallel Neural Local Lossless Compression"

By Mingtian Zhang, James Townsend, Ning Kang, David Barber

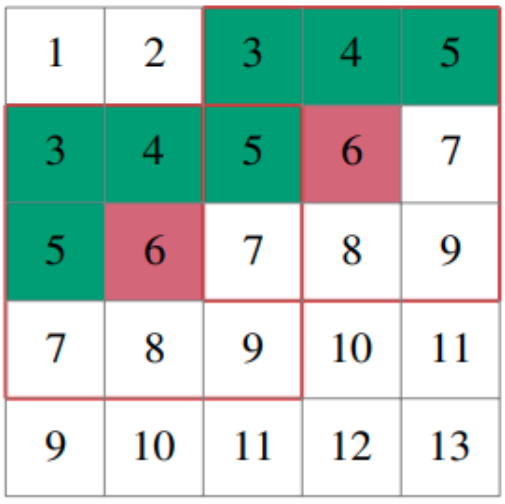


Figure 2.6 - Parallel Neural Local Lossless Compression

Summary:

Based on a local autoregressive model, the recently suggested “Neural Local Lossless Compression” (NeLLoC) has attained "state-of-the-art (SOTA) out-of-distribution (OOD)" generalization performance in the image compression test. Aside from encouraging OOD generalization, the native model also facilitates concurrent inference during the decoding step. The authors suggested a parallelization approach for localized autoregressive models in this study. They examine the challenges of implementing this technique and present experimental proof of considerable compression runtime improvements over the old, non-parallel implementation.

7. “Pseudocylindrical Convolutions for Learned Omnidirectional Image Compression”

By Mu Li, Kede Ma, Jinxing Li, David Zhang

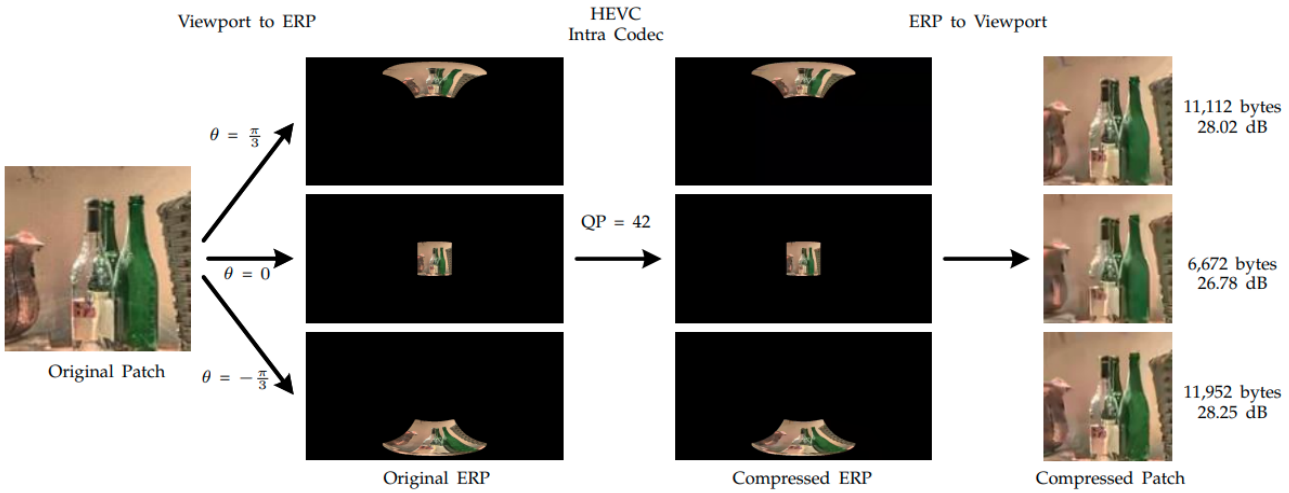


Figure 2.7 - Pseudocylindrical Convolutions for Learned Omnidirectional Image Compression

Summary:

Though "equirectangular projection" (ERP) is a handy way to store omnidirectional pictures (often referred as 360º images), this is neither equatable nor conformal, making future visual communication difficult. ERP would over-sample and warp items and stuff at the poles in the context of picture compression, making perceptually optimum bit allocation difficult. Methods such as zone packing and tiling representation are used in traditional 360-degree picture compression to mitigate the over-sampling problem, with little effectiveness. The

authors conduct one of the first efforts to form dnn for omnidirectional picture compression in this research. They begin by introducing parameterized pseudocylindrical notation as a generalization of ordinary pseudocylindrical map projections. To identify the (sub)-optimal design of the pseudocylindrical model in terms of a novel proxy goal for rate-distortion performance, a computationally efficient greedy technique is proposed. Then, for 360-degree picture compression, they suggest pseudocylindrical convolutions. Under suitable parametric restrictions, the pseudocylindrical convolution may be easily performed using normal convolution with so-called pseudocylindrical padding. They develop an end-to-end 360-degree picture compression system, consisting of the learnt pseudocylindrical notation, an evaluation transform, a quasi quantizer, a synthesis convert, and an entropy model, to illustrate the practicality of their approach. Experiment findings on 19,790 omnidirectional photos reveal that their technique regularly outperforms rival methods in terms of rate-distortion performance. Furthermore, their technology dramatically improves visual quality for all pictures at all bitrates.

8. "Towards End-to-End Image Compression and Analysis with Transformers"

by Yuanchao Bai, Xu Yang, Xianming Liu, Junjun Jiang, YaoWei Wang, Xiangyang Ji, Wen Gao

(a) JPEG (b) JPEG2000 (c) BPG

(d) mbt-m (e) mbt-m (JFT) (f) Ours

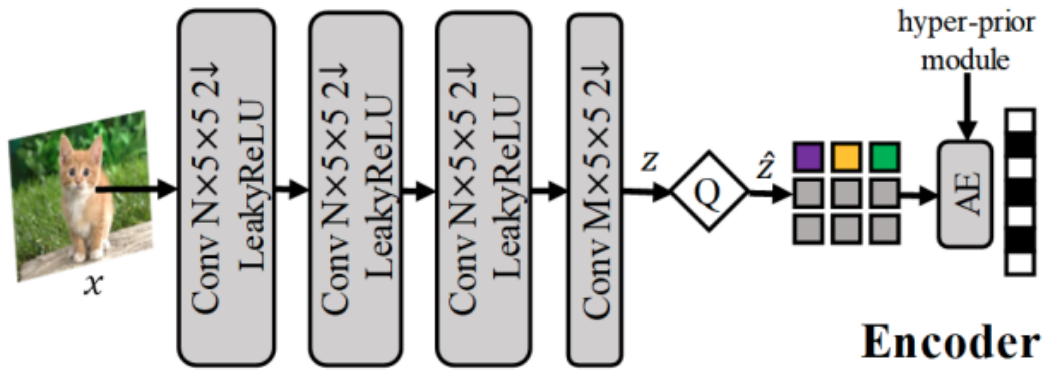


Figure 2.8 - Towards End-to-End Image Compression and Analysis with Transformers

Summary:

The authors present a Transformer-based end-to-end picture compressing and analysis model aimed at cloud-based image categorization applications. Instead of directly following an image codec with a current Transformer-based image classification model, they intend to redevelop the "Vision Transformer" (ViT) method to optimize image classification from compressed characteristics and facilitate image compression with long-term details from the Transformer.

They begin by replacing the ViT model's patchify stem (i.e., picture splitting and embedding) with a compact image encoder modeled by a cnns. The image encoder's compressed features are injected with neural inductive bias and then sent to the Transformers for classification tasks, avoiding picture reconstruction. In the meanwhile, they suggest a feature aggregation module that will merge the compact features with the

Transformer's selected intermediate features and send the combined aspects toward a deconvolutional cnn architecture for picture reconstruction. The consolidated features can gather long-term feedback from the Transformer's self-attention mechanism and increase compression performance. A two-step training technique is eventually used to tackle the rate-distortion-accuracy optimization problem.

The experimental findings show that the suggested model is successful in both picture compression and classification tasks.

9. “DPICT: Deep Progressive Image Compression Using Trit-Planes”

By Jae-Han Lee, Seungmin Jeon, Kwang Pyo Choi, Youngo Park, Chang-Su Kim

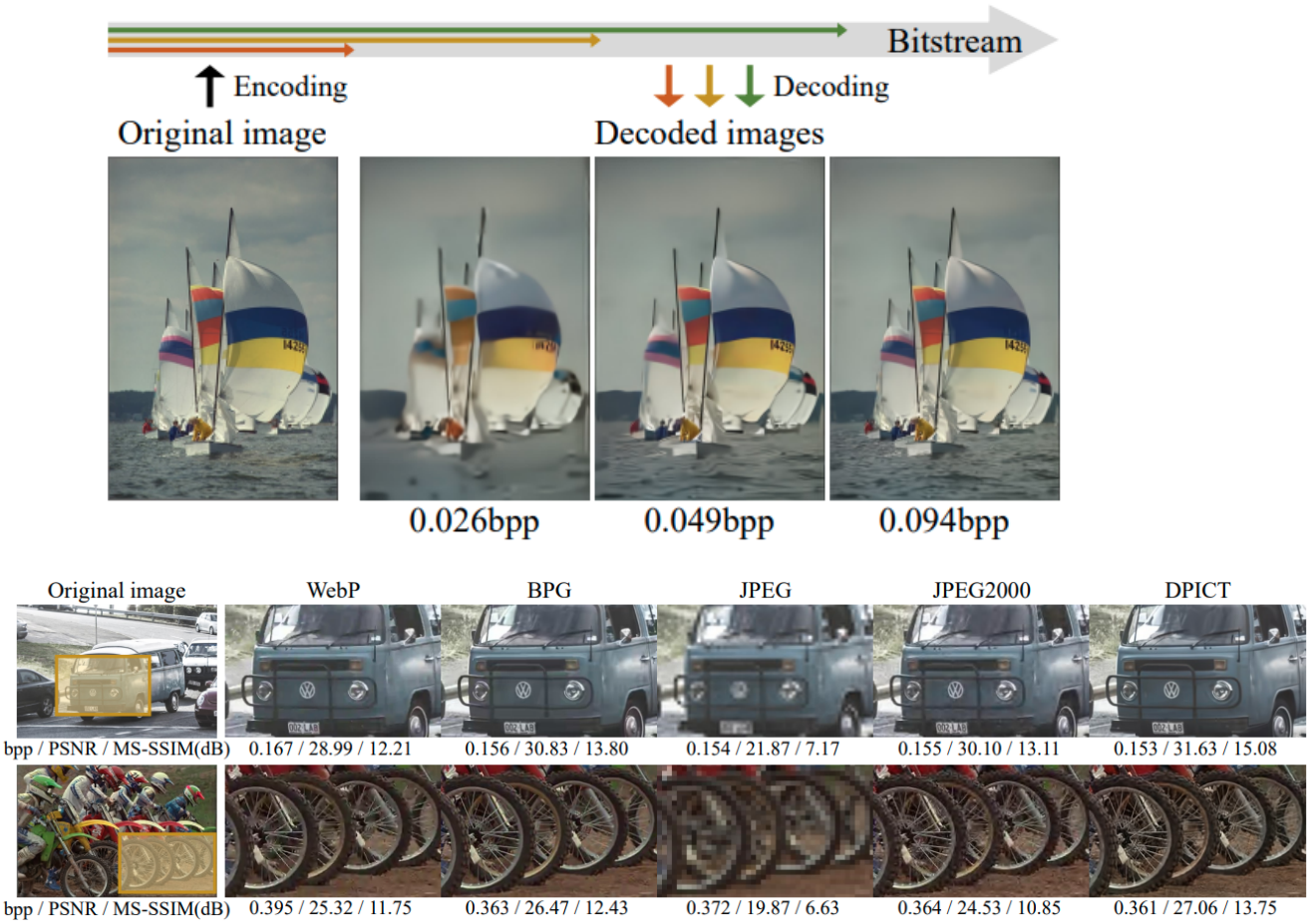


Figure 2.9 - DPICT: Deep Progressive Image Compression Using Trit-Planes

Summary:

The authors present the deep sequential image compression with trit-planes (DPICT) technique, the very first learning-based codec with "fine granular scalability" (FGS). First, they use an analysis network to convert a picture into a latent tensor. The latent tensor is then represented in triple digits (trits) and encoded trit-plane by trit-plane in decreasing order of importance into a compressed bitstream. Furthermore, inside each trait-plane, the traits are sorted as per their rate-distortion priorities, with the most critical information transmitted first. Because the compressing network is less optimal for scenarios when fewer trit-planes are used, they create a postprocessing network to refine reconstructed pictures at low rates. DPICT greatly surpasses standard progressive codecs in terms of performance while permitting FGS transmission, according to experimental data.

10. “End-to-end optimized image compression with competition of prior distributions”

by Benoit Brummer, Christophe De Vleeschouwer

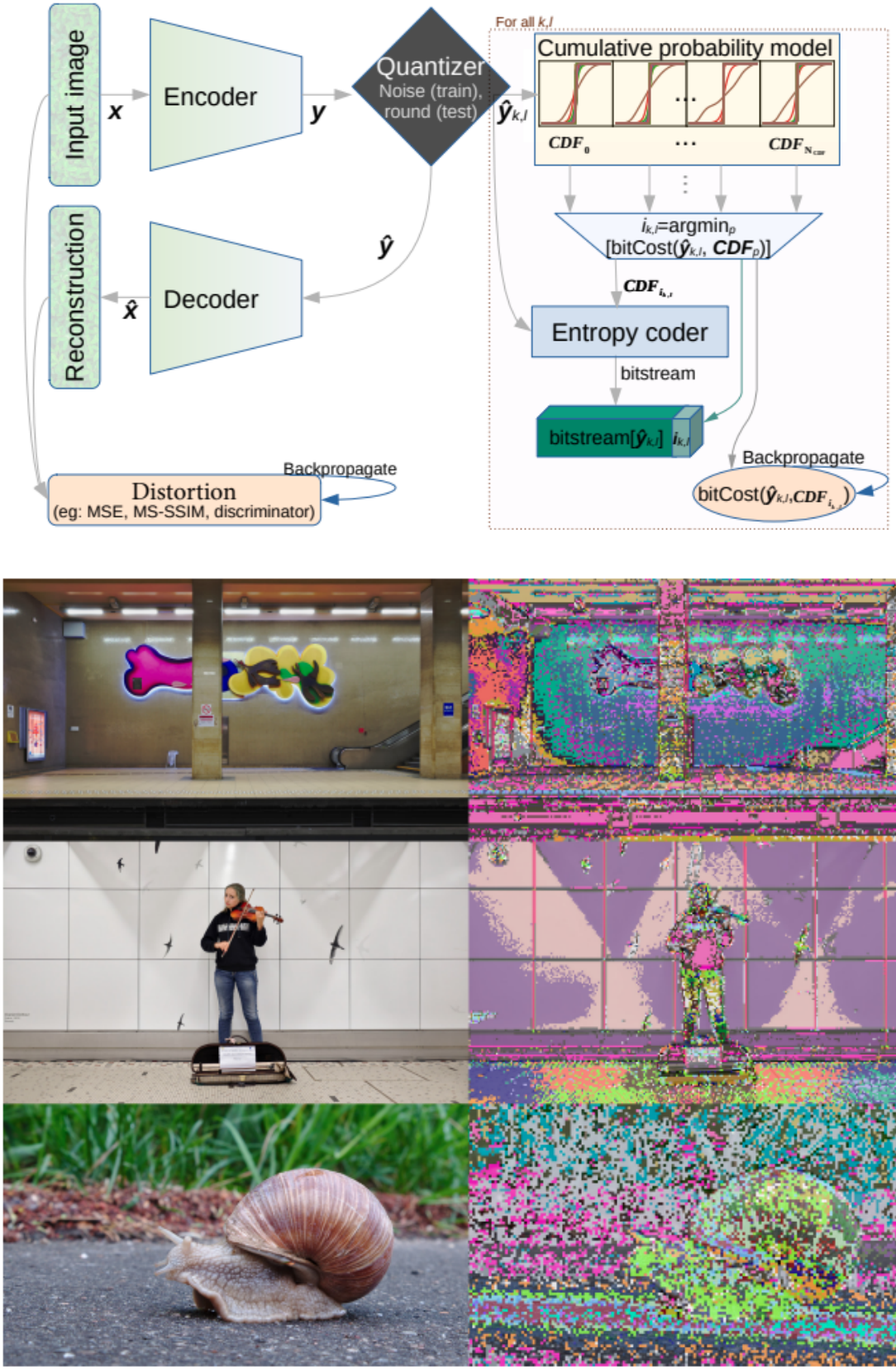


Figure 2.10 - End-to-end optimized image compression with competition of prior distributions

Summary:

Convolutional autoencoders have risen to the top of picture compression research. Encoder output is generally processed using a second autoencoder to build per-variable customized prior probability distributions to optimize entropy coding. Instead, the authors suggest a compression strategy that uses a typical cnn autoencoder and numerous learned prior distributions to compete with experts. A static database of cumulative probability functions stores trained past distributions. During inference, an entropy coder uses this table as a look-up table to establish the optimum prior for every spatial point. Their solution achieves rate-distortion performance equivalent to a forecasted parametrized prior while requiring a fraction of the entropy coding and decoding effort.

11. "Quantum pixel representations and compression for N-dimensional images"

By Mercy G. Amankwah, Daan Camps, E. Wes Bethel, Roel Van Beeumen, Talita Perciano

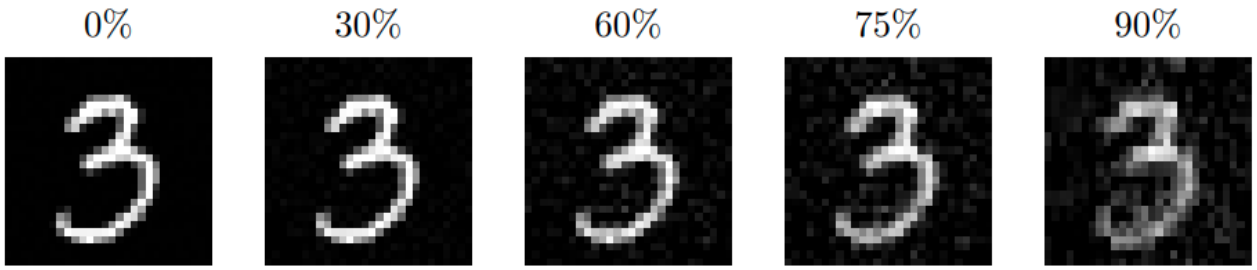


Figure 2.11 - Quantum pixel representations and compression for N-dimensional images

Summary:

The authors describe an innovative and consistent framework for quantum image representations that encompasses several of the most famous representations presented in recent literature, including "(I)FRQI, (I)NEQR, MCRQI, and (I)NCQI." The proposed QPIXL paradigm leads to more efficient network implementations and dramatically lowers gate complexity for all quantum pixel representations investigated. Their technique uses no ancilla qubits and simply requires a linear amount of gateways in terms of total quantity of pixels. Furthermore, because the circuits only use Ry gates and CNOT gates, they are practicable in the NISQ period. Furthermore, they present a network and image compression technique that has been demonstrated to be extremely successful, capable of reducing the

number of gates required to produce a FRQI state for example, scientific photos by up to 90% without affecting image quality. The algorithms are made public as part of the Quantum Image Pixel Library, QPIXL++.

12. "Towards Flexible Blind JPEG Artifacts Removal"

By Jiaxi Jiang, Kai Zhang, Radu Timofte

Figure 2.12 - Towards Flexible Blind JPEG Artifacts Removal

Summary:

Because of its ease of use, developing a single hidden layer blind network to handle many quality variables for JPEG picture artifact removal has received a lot of interest. However, present deep blind approaches often rebuild the picture without anticipating the quality factor, limiting their ability to modify the output in the same way as non-blind methods do. To address this issue, the authors suggest FB"CNN, a customizable blind dnn that can anticipate the configurable quality factor to govern the trade-off between artifact removal and detail retention. Specifically, for flexible control, FB"CNN" separates the data rate from the JPEG picture using a decoupler component and then embeds the anticipated data rate

into the succeeding reconstructor module using a parameter attention block. Furthermore, they discover that previous approaches failed on non-aligned dual JPEG pictures even with a one-pixel displacement, and they suggest a dual JPEG degeneration model to supplement the training data. Extensive testing on single JPEG photos, more generic twin JPEG pictures, and authentic JPEG images show that their proposed FB "CNN" outperforms state-of-the-art approaches in regard to both quantitative measures and visual quality.

13. **"Image Compression with Recurrent Neural Network and Generalized Divisive Normalization"**

By Khawar Islam, L. Minh Dang, Sujin Lee, Hyeonjoon Moon

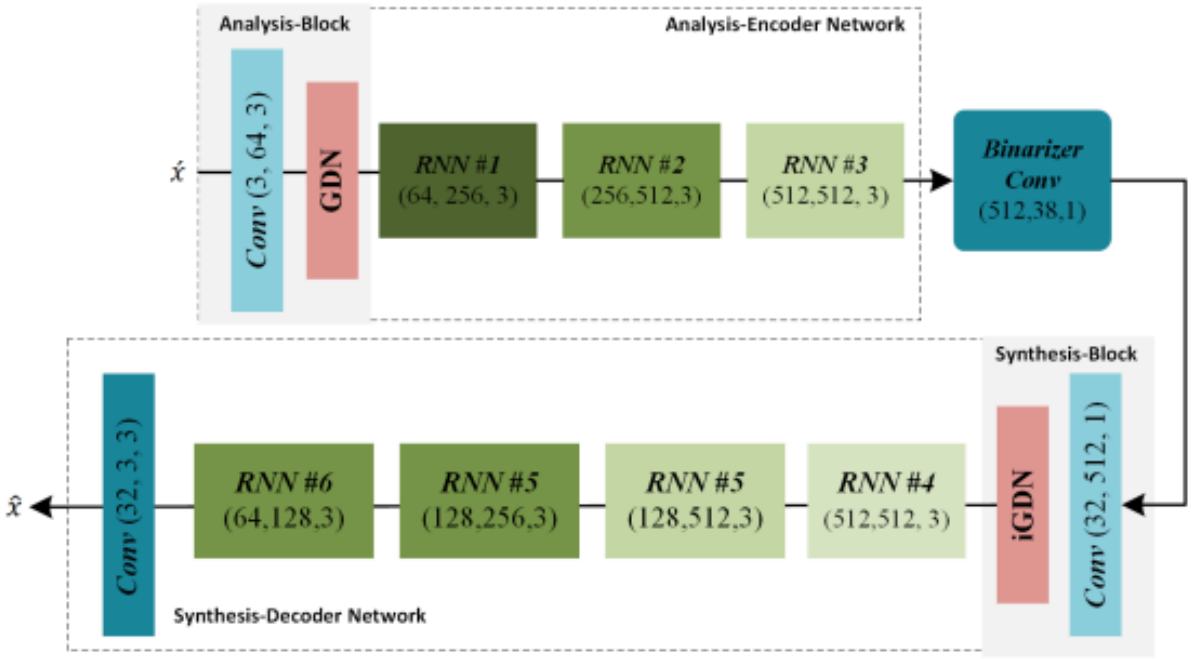


Figure 2.13 - Image Compression with Recurrent Neural Network and Generalized Divisive Normalization

Summary:

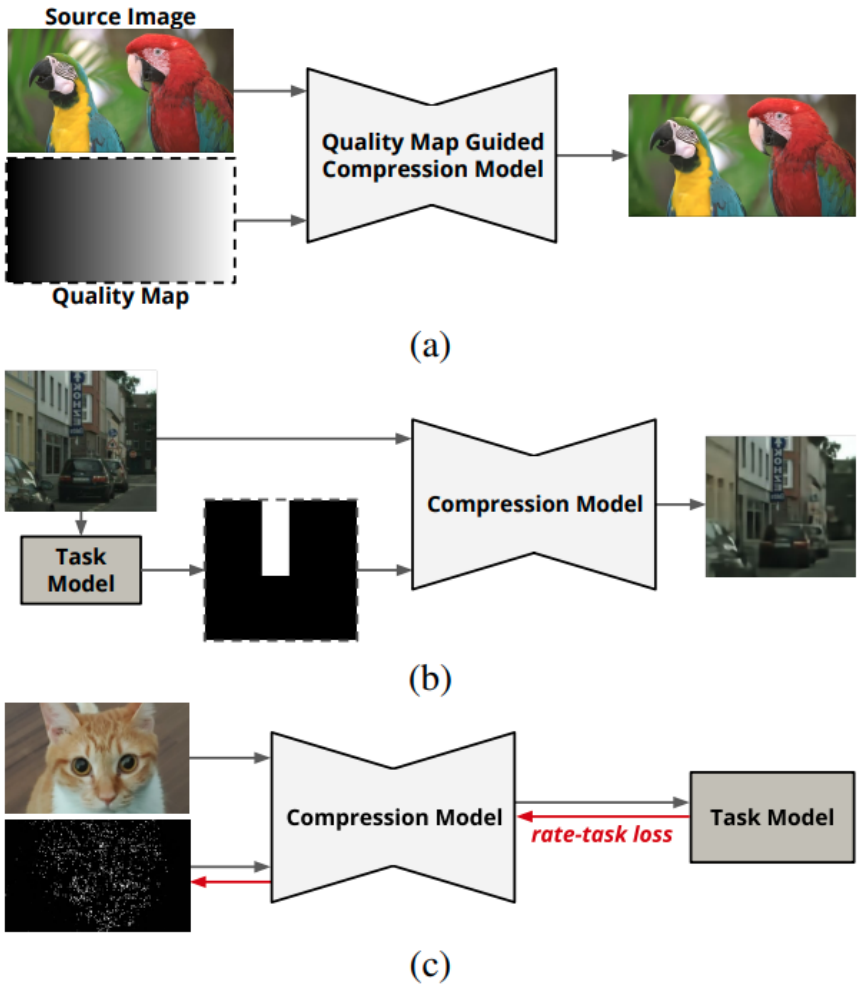
Image compression is a technique for eliminating spatial redundancy between neighboring pixels in order to recreate a high-quality image. Deep learning has received a lot of attention from the scientific community in recent years, and it has yielded promising picture reconstruction outcomes. As a result, current strategies have concentrated on building

broader and more meaningful networks, considerably increasing network complexity. In this study, two successful innovative blocks are developed: an evaluation and synthesis block that uses the hidden layers and "Generic Divisive Normalization (GDN)" in the changeable encoder and decoder sides, and a changeable encoder and decoder side. For quantization, their network uses a pixel RNN technique.

Furthermore, in order to enhance the overall system, the researchers encode a residual picture utilizing LSTM cells to remove extraneous information. In terms of picture similarity, experimental findings show that the recommended variable-rate framework with innovative blocks outperforms known approaches and standard image codecs like as George's cite002 and JPEG.

14. "Variable-Rate Deep Image Compression through Spatially-Adaptive Feature Transform"

By Myungseo Song, Jinyoung Choi, Bohyung Han



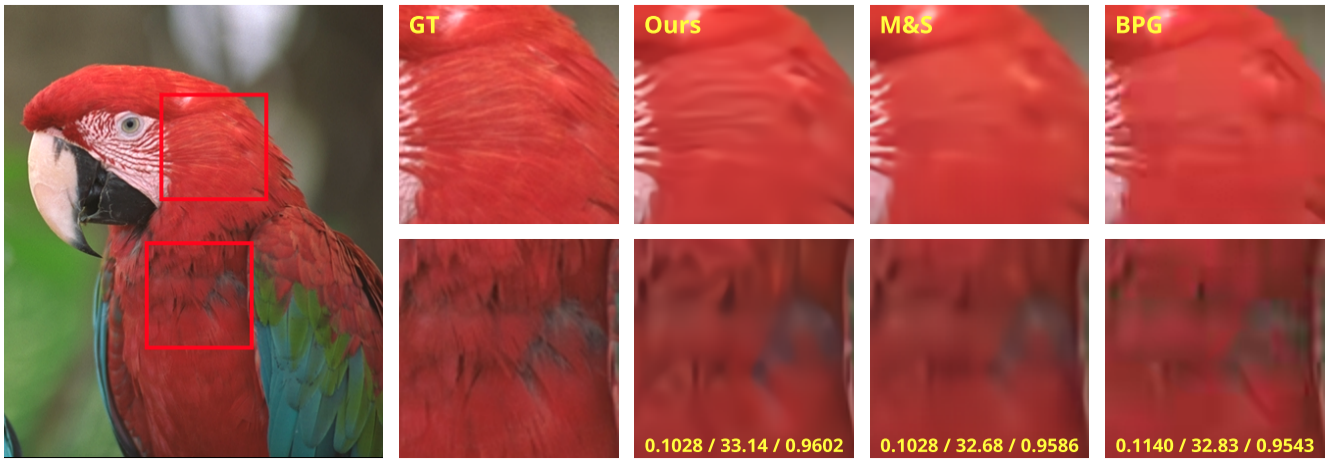


Figure 2.14 - Variable-Rate Deep Image Compression through Spatially-Adaptive Feature Transform

Summary:

The authors present a flexible deep image compression system based on the "Spatial Feature Transform" (SFT arXiv:1804.02815), that takes as inputs a source picture and a matching quality map and outputs a small file size with varying rates.

Their methodology uses a single model to span a wide variety of compression rates, that is handled by variable pixel-wise quality mappings.

Furthermore, the proposed framework enables us to execute task-aware photo compressions for a variety of tasks, such as classification, by effectively calculating optimum quality maps for the encoding network. This is even achievable using a pretrained network, as opposed to learning different models for each job.

When compared to alternatives based on numerous models that are tuned independently for several distinct target rates, their algorithm achieves an amazing rate-distortion trade-off.

Without further model training, the suggested strategy successfully increases performance on picture classification and text area quality preservation at the same degree of compression.

15. “Enhanced Invertible Encoding for Learned Image Compression”

By Yueqi Xie, Ka Leong Cheng, Qifeng Chen

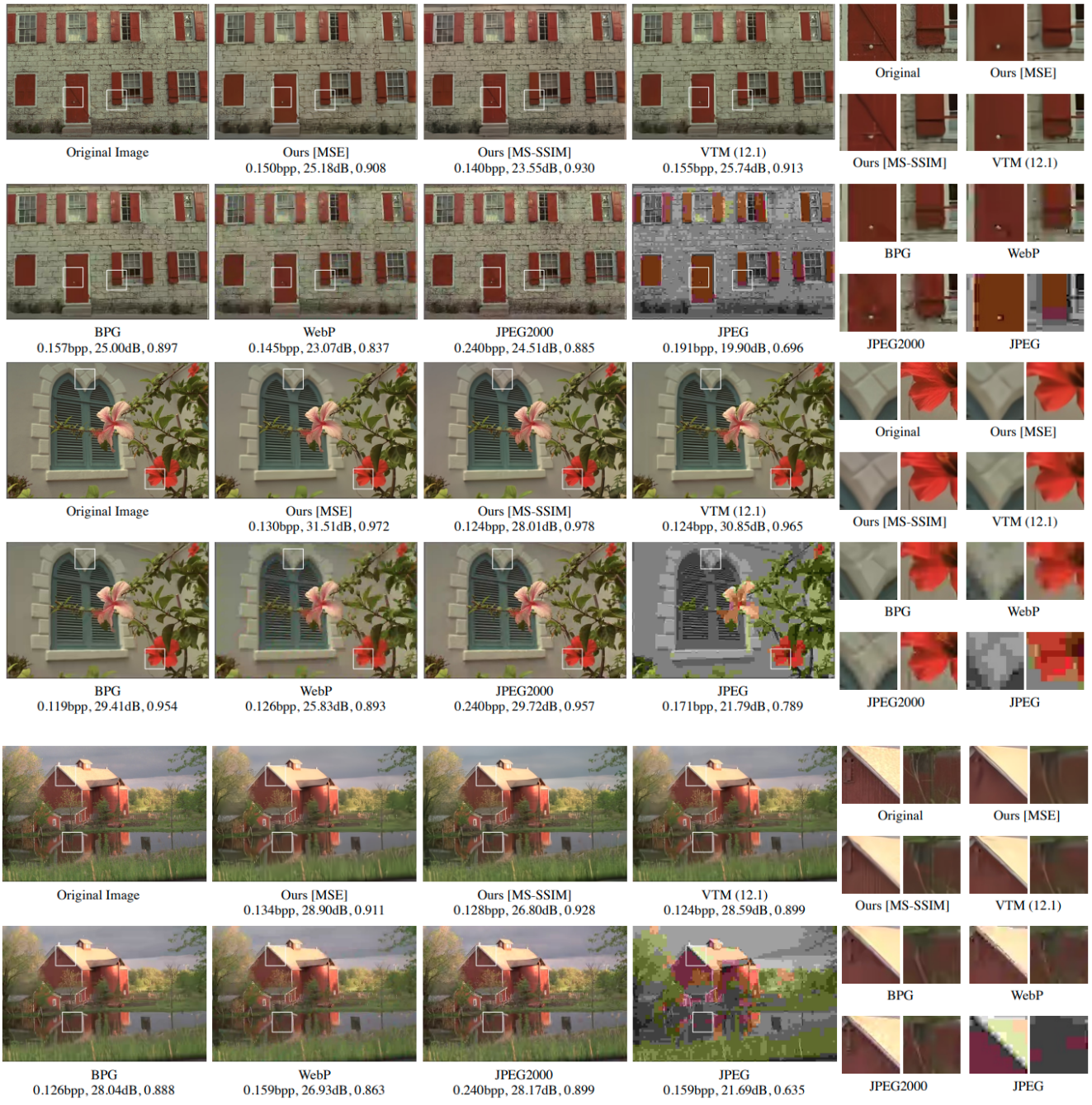


Figure 2.15 - Enhanced Invertible Encoding for Learned Image Compression

Summary:

Although deep learning-based image compression systems have made significant progress in recent years, their performance still falls short of the newest reduction benchmark "Versatile Video Coding (VVC)." The majority of recent improvements have been on developing a much more realistic and versatile entropy model capable of properly

parameterizing the distributions of latent characteristics. However, limited attempts are being made to design a better transition between image space and latent feature space. Instead of using prior autoencoder type networks to generate this transformation, the authors present an upgraded Invertible Encoding Network utilizing "invertible neural networks (INNs)" to greatly reduce information loss for greater compression. Experiment findings mostly on Kodak, CLIC, and Tecnick collections demonstrate that their technique surpasses existing learning image compression algorithms and compression standards, such as VVC (VTM 12.1), particularly for high-resolution photos.

2.2 Integrated summary of research papers

Image compression is among the most elementary techniques and applications in the world of image and video processing. Earlier approaches created a very well defined pipeline, and attempts were made to optimize all pipeline modules through customized tweaking. Later, enormous contributions were made, particularly when data-driven approaches reinvigorated the area with their superior modeling capabilities and flexibility in adding newly built modules and restrictions. Despite significant development, there is a lack of systematic standards and complete study of end-to-end learnt picture compression algorithms.

Convolutional neural networks ("CNNs") outperform classic computer vision models in terms of compression artifact reduction and super-resolution performance. "CNN" convolution processes enable them to discover how surrounding pixels correlate. The features of complex pictures are mirrored by cascaded convolution procedures. However, using a "CNN" network all through the jpeg compression process is difficult since it needs gradient descent methods and backpropagation that are difficult to include in end-to-end picture compression.

Image compression based on "CNN" has higher "Peak Signal-to-Noise Ratio (PSNR)" and "Structural Similarity (SSIM)" as compared to JPEG2000. RNN based image compression methods have also been used in models which have shown good performance. This resulted in picture compression performance levels nearing High-Efficiency Video Coding (HEVC).

Chapter 3: Requirement Analysis and Solution Approach

3.1 Overall description of the project

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. Image compression deals with minimizing the number of bits required to represent an image.

3.2 Requirement Analysis

For Image Compression, we will be using Deep Learning. And so for that Neural Networks and Convolutional Neural Networks and GANs (Generative Adversarial Networks) will be used. This requires a good amount of computational power and GPU power to do the required tasks quickly and efficiently.

We require the following for the project:-

- A laptop or computer running Windows 10 or macOS Sierra or above or Linux, or Ubuntu 16.04 LTS
- A dual core processor 2 GHz or more
- 8GB of RAM or more
- NVidia GPU 1000 series or above
- Jupyter Notebook
- Google Colab

We also require these things:-

● For training our model we need training data. For this we need a large dataset of a wide variety of images. We need this, so that the model can learn from it and is able to generalize well and compress the images efficiently. It is also needed for checking the PSNR ratio of original and compressed images and so that the Discriminator can learn from it.

● For validation purposes, we also need testing data, so that we can check the performance of our model and fine tune it and see how well it can compress images and create images that are similar to the input image.

3.2.1 Datasets used

Our training set consists of a large set of high-resolution images collected from the Internet. We evaluated on four diverse benchmark datasets collected to demonstrate that our method generalizes beyond the training as well: the widely used Kodak dataset, as well as the CLIC2020 dataset and DIV2K dataset along with a subset of the Imagenet Dataset.

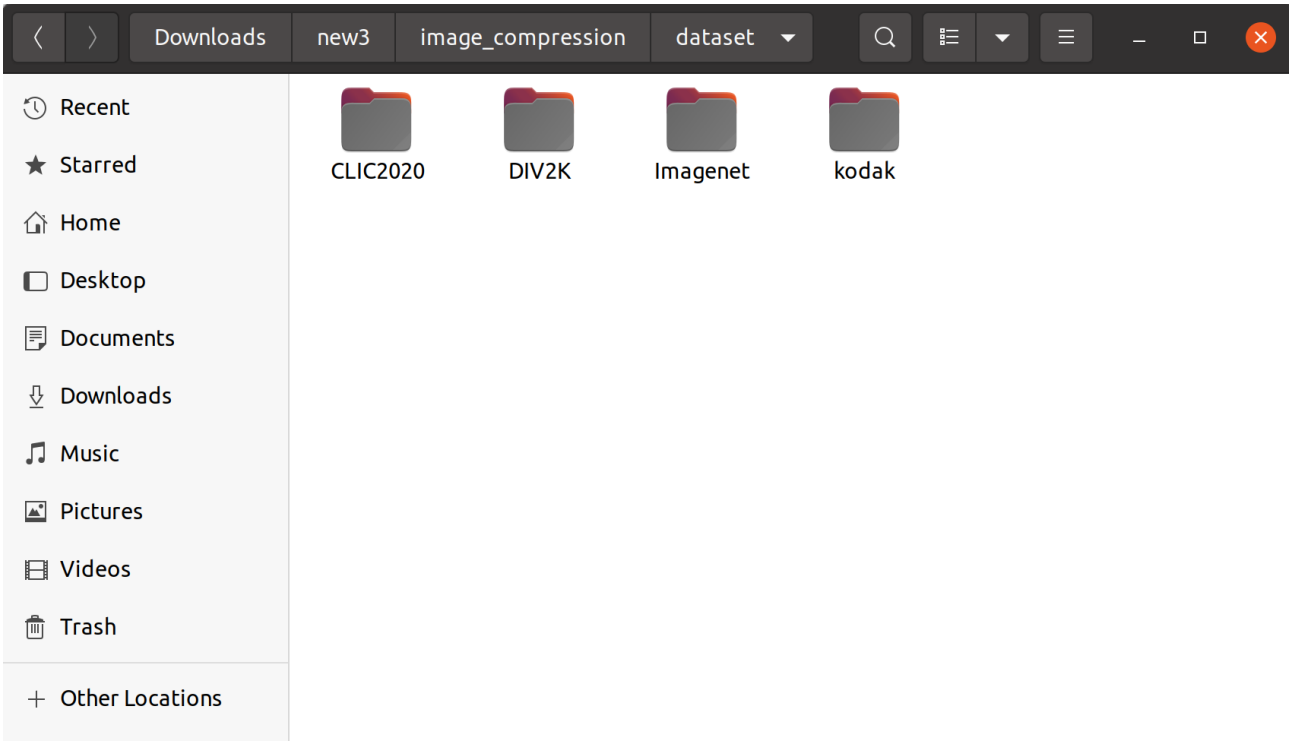


Figure 3.1 - All four Datasets used by us for training and testing our model

Kodak Dataset:

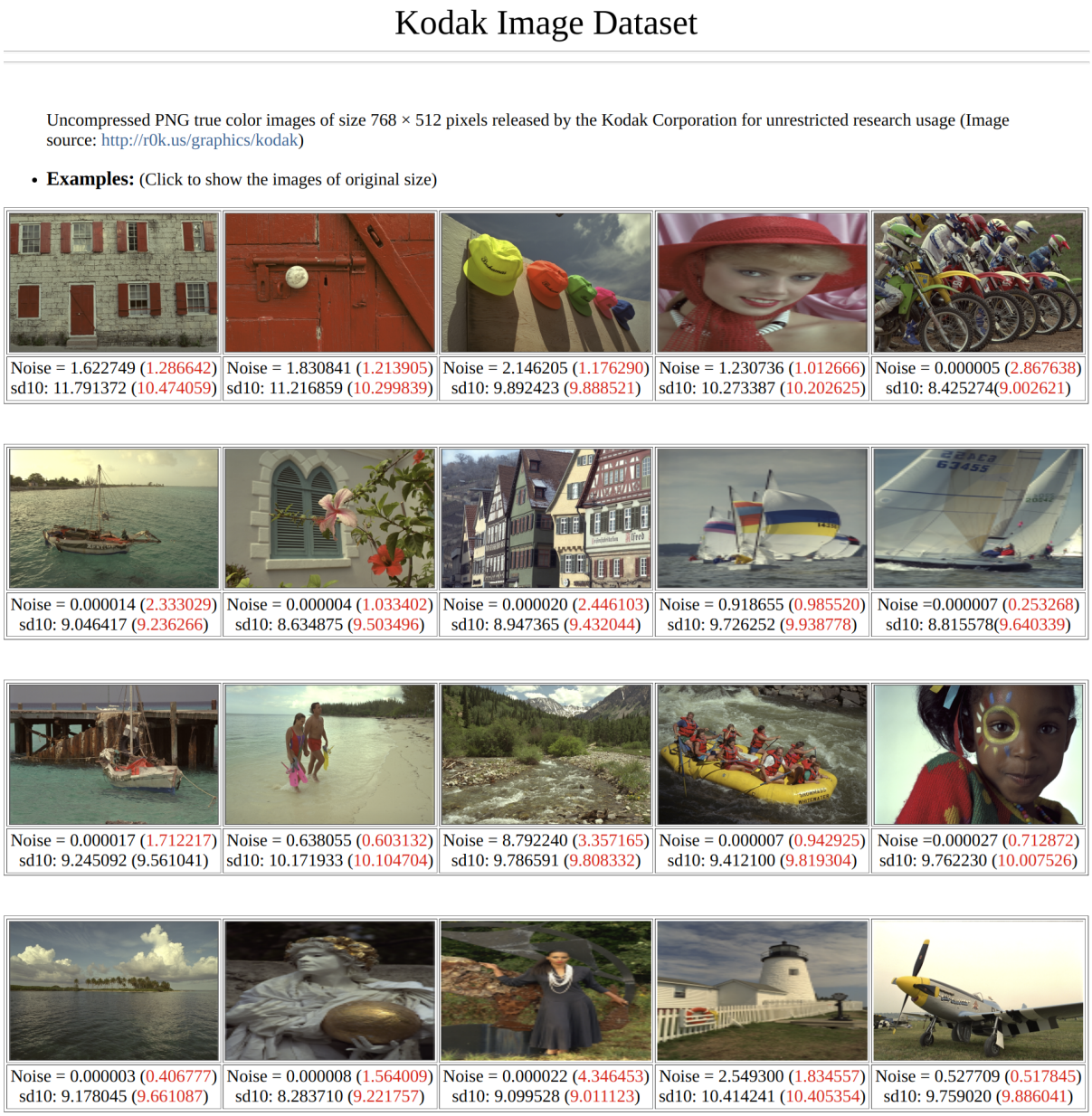


Figure 3.2 - The Kodak Dataset

Kodak Dataset is a collection of uncompressed PNG images provided by the Kodak Corporation for Machine Learning and Computer Vision research use. These are high resolution images of various subjects which will be used for training and testing our model.

CLIC2020 Dataset:

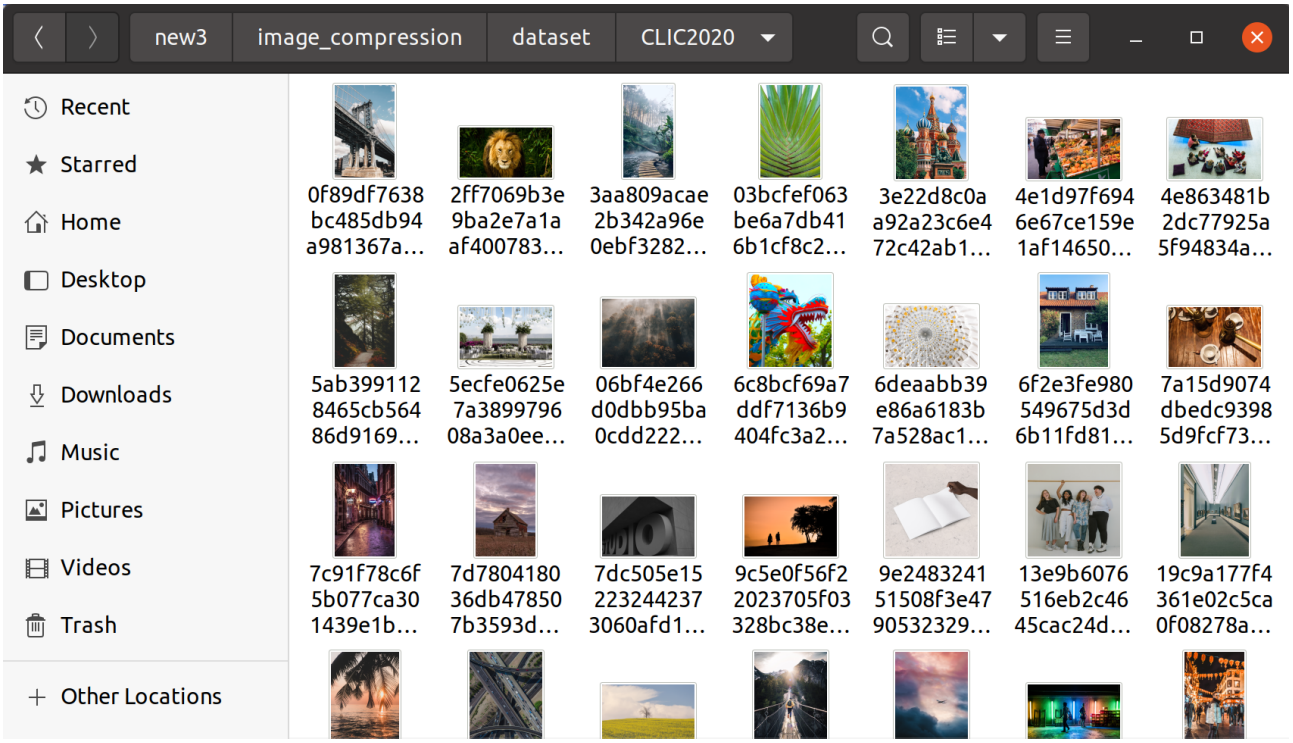


Figure 3.3 - The CLIC2020 Dataset

It is a dataset of a wide variety of high resolution images provided by Workshop and Challenge on Learned Image Compression 2020, so that the participants can use them to train and test their image.

DIV2K Dataset:

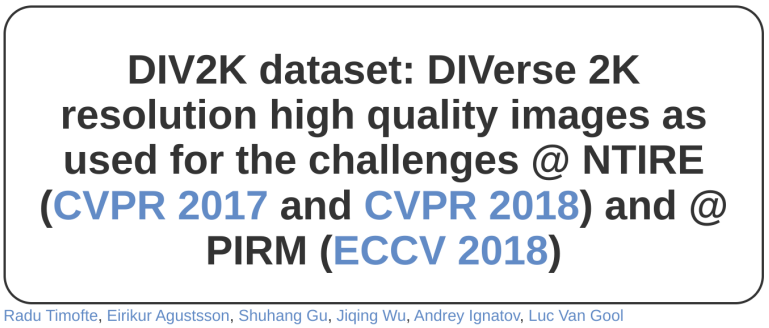


Figure 3.4 - The DIV2K Dataset

DIV2K is a famous dataset containing training, testing and validation data of around 1700 images which we have used to train our model and test our model's performance. It contains images from a wide variety of scenarios.

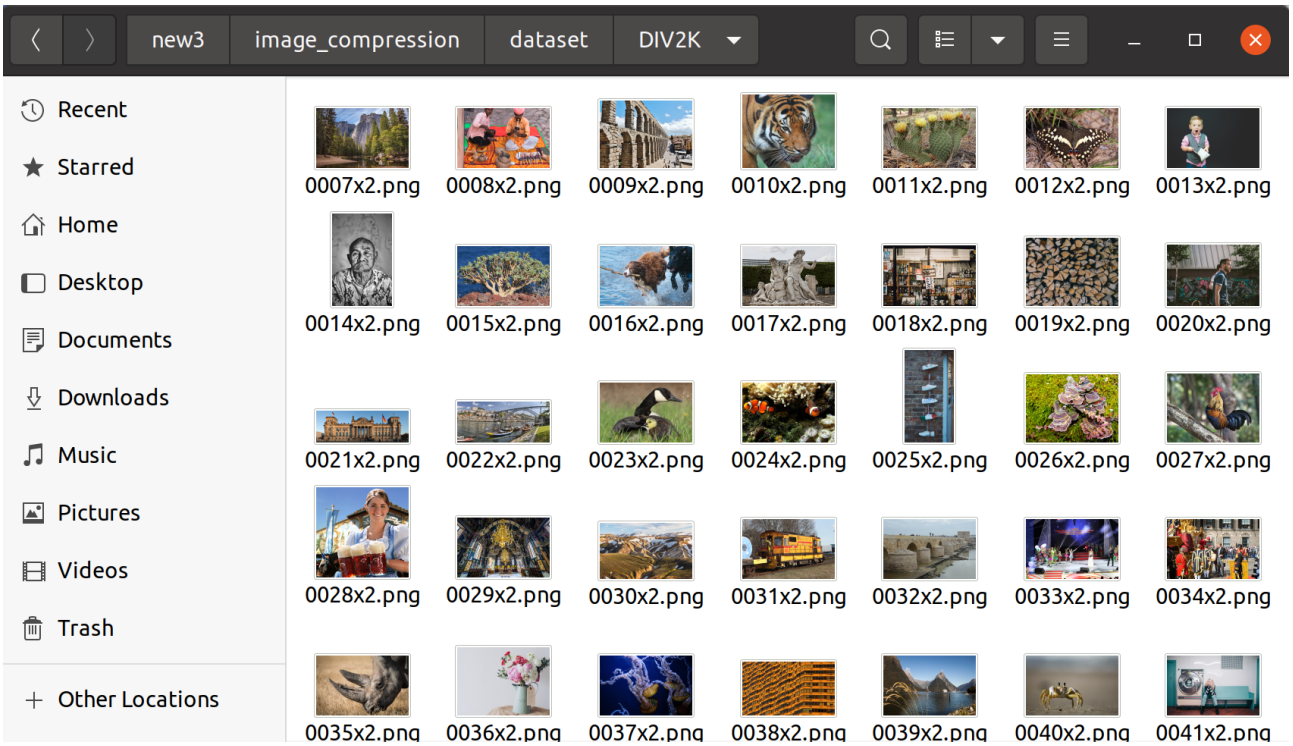


Figure 3.5 - The DIV2K Dataset

ImageNet Dataset:

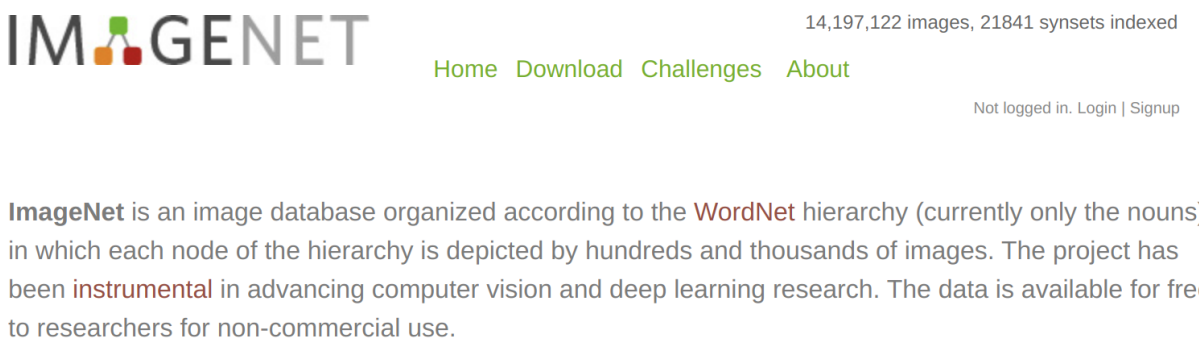


Figure 3.6 - The ImageNet Dataset

We have used a subset of the immensely popular ImageNet dataset provided by the Stanford Vision Lab for our model. Our subset of the ImageNet dataset contains 9,650 high resolution images covering different animals, landscapes, buildings and scenes, so that our model can generalize well.

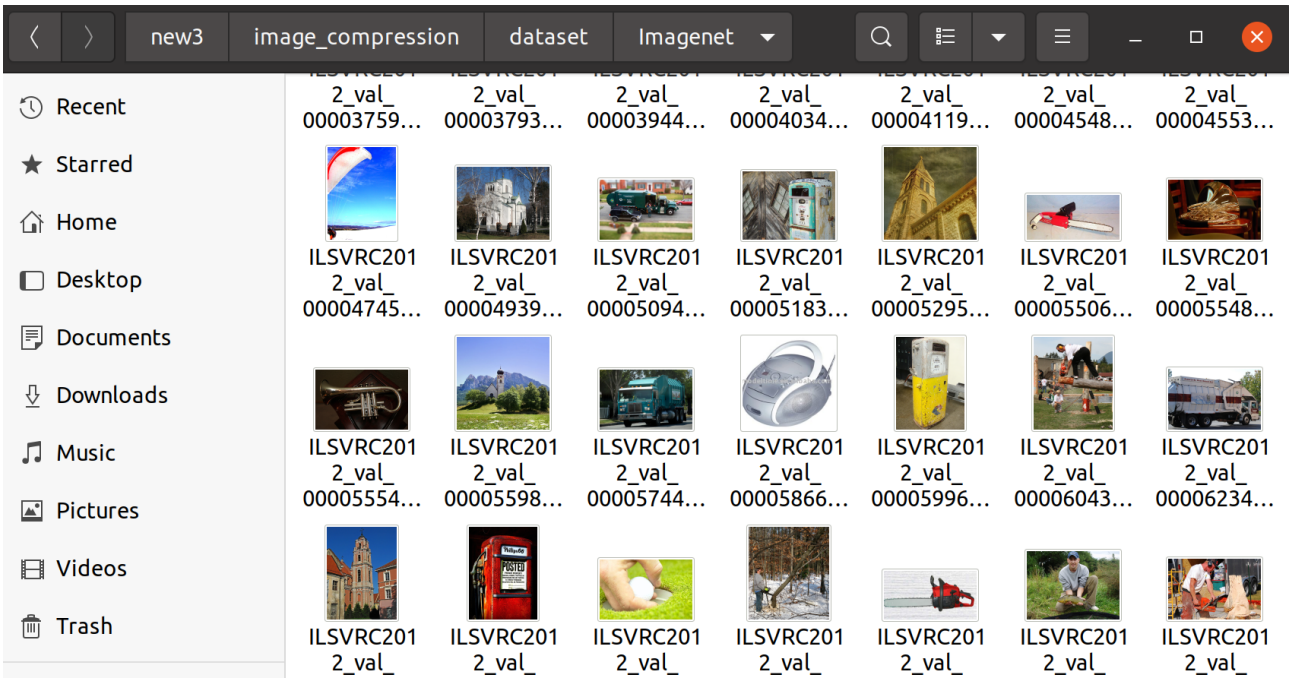


Figure 3.7 - The ImageNet Dataset

We divided all the collected datasets into training and testing data in the ratio 80% to 20%. This is important to be done, so that the model can learn and get trained on the input images.

3.2.2 Technologies Used

1. Programming Language: Python

Figure 3.8 - Python Programming Language

Python is an interpreted and high-level programming language. It is an object oriented programming language. It is simple and has easy to learn syntax. It is incredibly fast as there is no compilation step. Python is the most used language for Machine Learning and Deep

Learning as it supports most libraries used in Machine Learning like scikit-learn, numpy, matplotlib, OpenCV, THeano, NLTK, Tensorflow and Pytorch.

2. Programming Framework:

a. PyTorch :

Figure 3.9 - PyTorch Library

PyTorch is an open source machine learning library for Python. It is used for computer vision and natural language processing. It is maintained and developed by FAIR (Facebook Artificial Intelligence Research Group).

Originally it was developed by Hugh Perkins as a Python wrapper for the LuaJIT. PyTorch implements Torch in Python while sharing the same core C libraries for the backend code.

Facebook Developers have tuned the back end code to run Python efficiently. Developers have kept the GPU based hardware acceleration as well as the extensibility features that made Lua-based Torch.

b. Tensorflow:

Figure 3.10 - TensorFlow Library

TensorFlow is a software library by Googlewhich is very famous and is used for implementing machine learning and deep learning models.

It includes a variety of machine learning and deep learning algorithms. It can train and run deep neural networks for many tasks like image recognition, handwritten digit classification and word embedding among others. It is also used for creation of various sequence models

3. Neural Networks:

a. Convolutional Neural Network (CNN) :

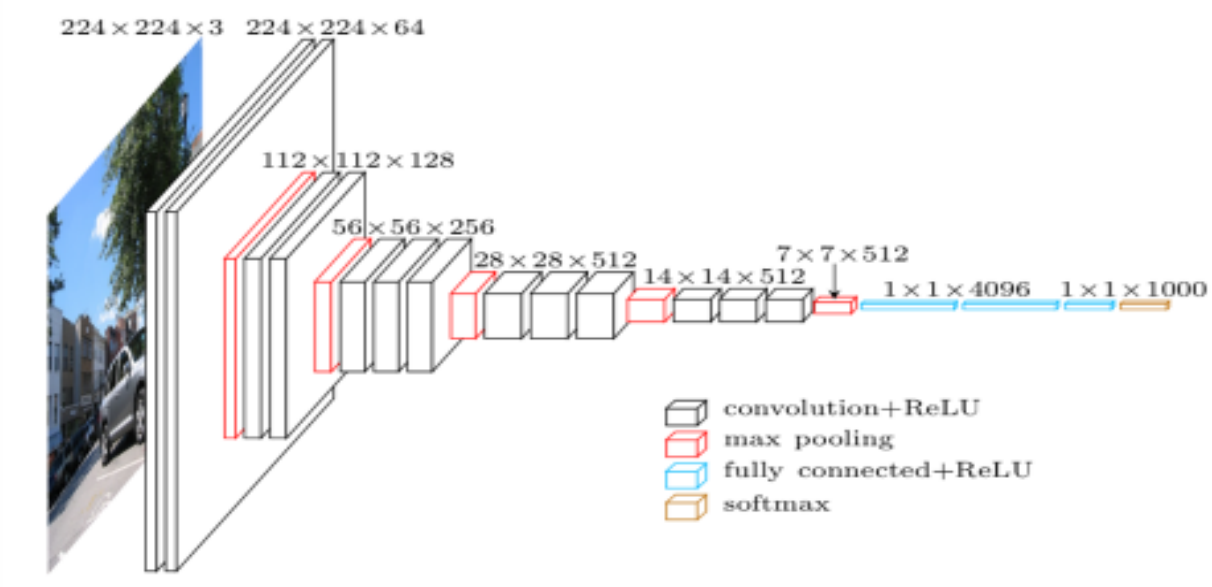


Figure 3.11 - Convolutional Neural Network

Convolutional neural networks perform great with image, speech, or audio signal inputs. They have three main types of layers-

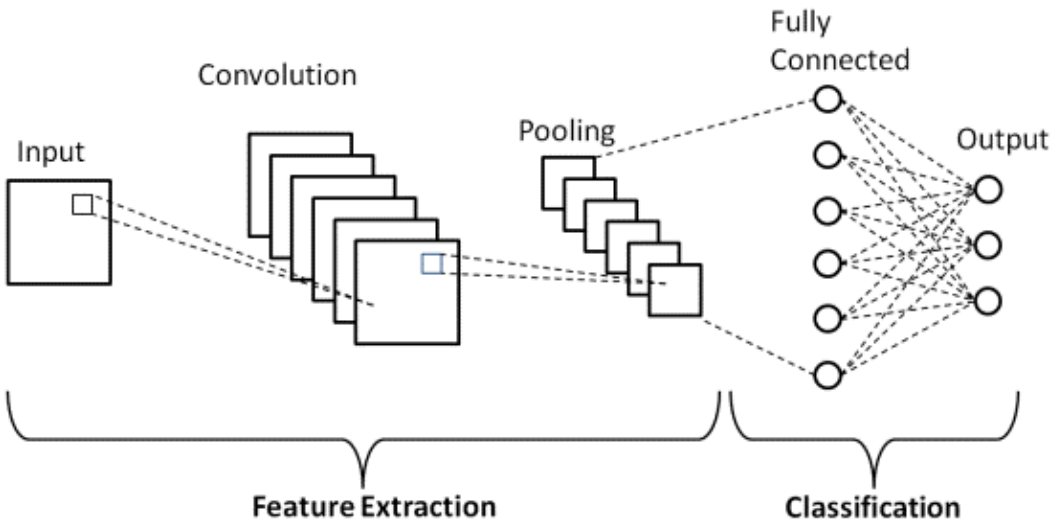


Figure 3.12 - Convolutional, Pooling and Fully Connected Layers

- Convolutional layer
- Pooling layer
- Fully Connected layer

The first layer in a convolutional network is the convolutional layer. Even though a convolutional layer can be followed by an additional convolutional layer or even a pooling layer, the final layer is the fully connected layer. Along with each layer, the complexity of the Convolutional Neural Network increases, with each next layer identifying greater portions of the image. The earlier layers usually focus on simple features like edges and colors for example. The layers towards the end, start to recognize larger shapes and elements of the object. This keeps going on until the convolutional layers finally identify the intended object.

b. Generative Adversarial Networks (GANs) :

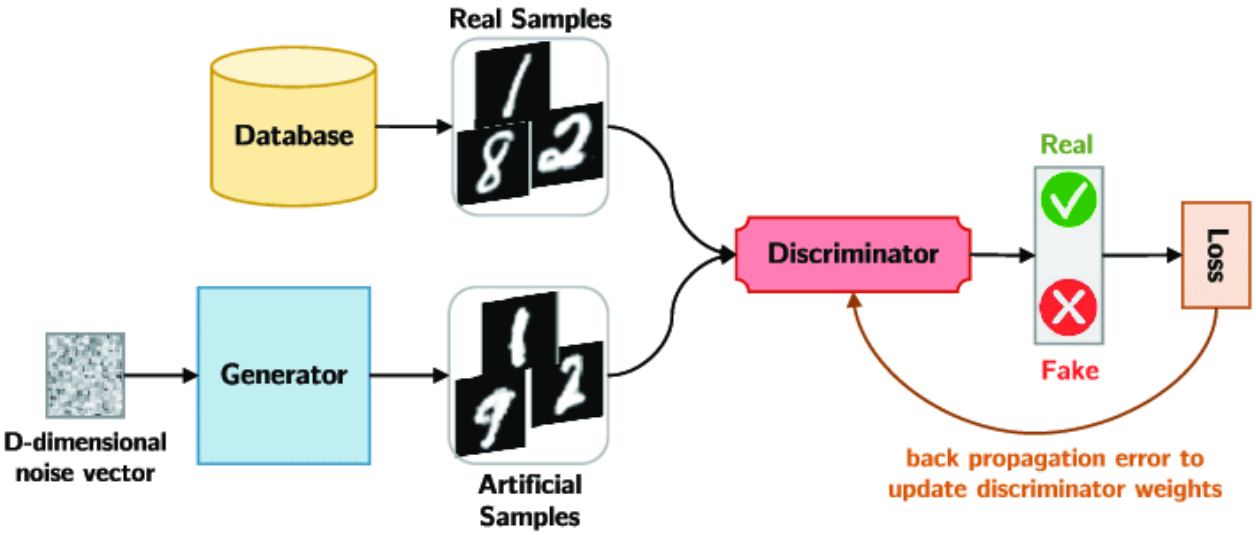


Figure 3.13 - Generative Adversarial Network (GAN)

The Generative Adversarial Network, consists of two parts -

The Generator and

The Discriminator

The Generator tries to maximize the probability of the Discriminator making a mistake and thinking that the image produced by the Generator is from the training set instead. This is how the Generator gets trained.

In the end, the Generator unknowingly gets trained and gets forced to create images which are very similar to the original images.

What Generator actually does is that it captures the data distribution of the image and creates new example images and the Discriminator estimates the probability that the example image generated from the Generator came from the training data rather than the Generator.

The discriminator is provided both with an input image which is real and the image generated by the Generator.

Discriminator classifies whether the image generated by the Generator is real or fake, and returns its assessment to the Generator, in turn teaching the generator to generate images which fool the discriminator.

5. IDE

a. Google Colab :

Figure 3.14 - Google Colab

Google Colab is a free Jupyter notebook environment which has the capability to run entirely in the cloud. It doesn't require any setup. Also, the notebooks which we create can be edited by our team members. It supports many popular computer vision and machine learning libraries which can be easily loaded in our notebook.

b. Jupyter Notebook

Figure 3.15 - Jupyter notebook

Jupyter Notebook is an open source web application which we use to create and also to share documents which contain text, equations, visualizations and documents. Actually, they are a spin off project from the main IPython (Interactive Python) project. The name, Jupyter, comes from - Julia, Python and R, which are the programming languages it supports. Moreover, it ships with the IPython kernel, which lets us write our own python programs.

3.3 Solution Approach | Proposed Solution

Overall Description of Proposed Solution:

Our proposed solution is to use GANs (Generative Adversarial Networks) and CNNs (Convolutional Neural Networks) for learned image compression. First Convolutional Neural Networks are fed the images from which they learn the important features in the image. Then they pass on these extracted features to the Generative Adversarial Network. In Generative Adversarial Network, 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.

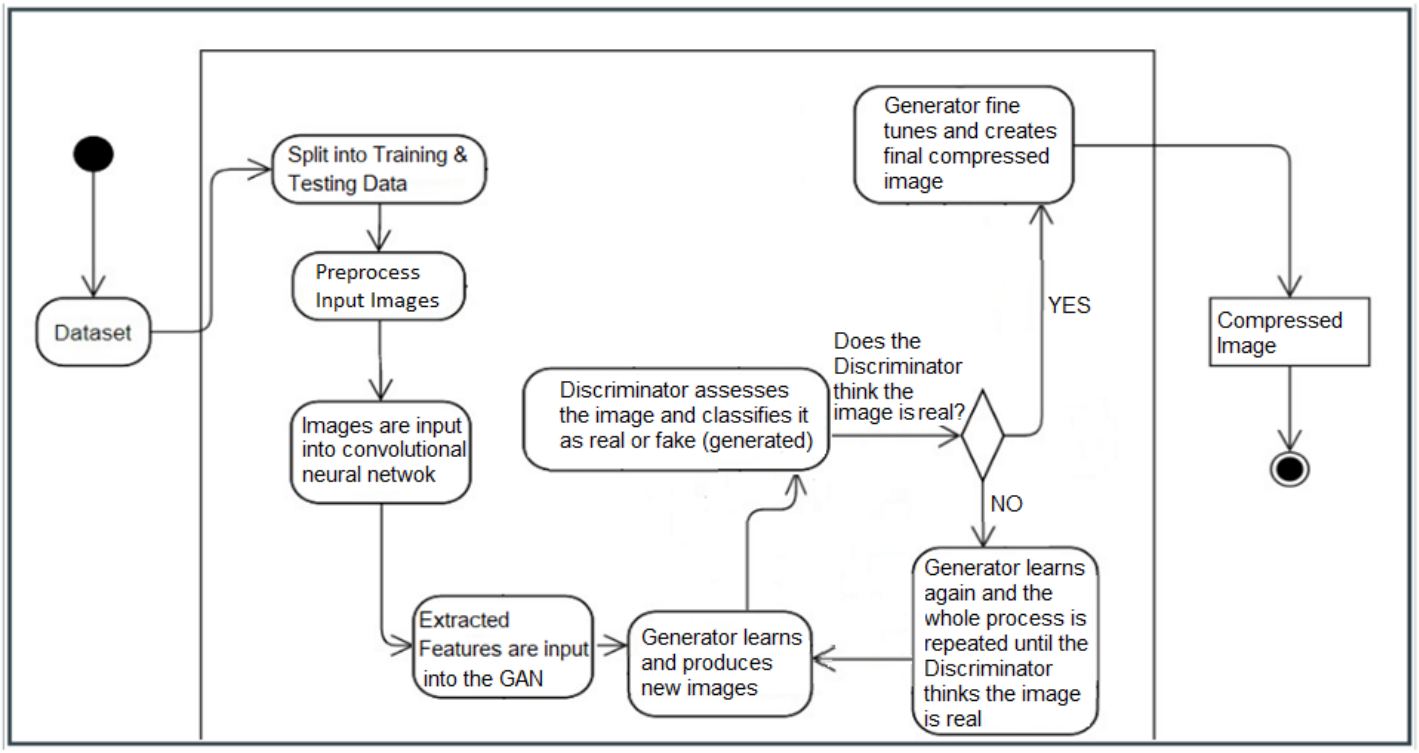


Figure 3.16 - Proposed Approach - Image Compression using Deep Learning

Our proposed solution consists of 5 steps:

1. **Feature Extraction from the images:**

   Our model needs to learn the features from the input images. For this, we use convolutional neural networks to extract features from the images. Now, our model starts learning from the images which are input into it. Moreover, in order to stop overfitting, we have added dropout layers between the convolutional layers. The output of these layers gives us the features our model has extracted from the images

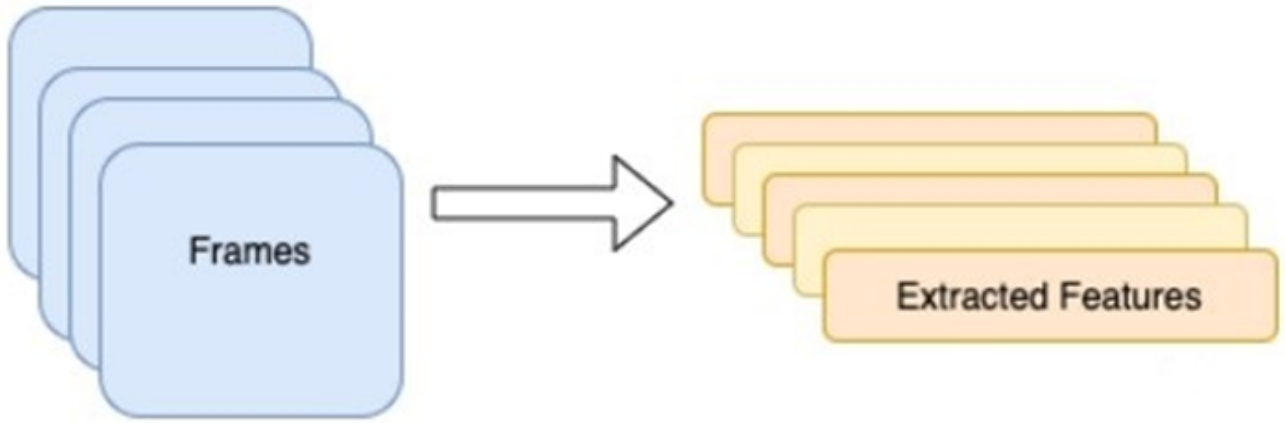


Figure 3.17 - Extracting Features from Image Frames

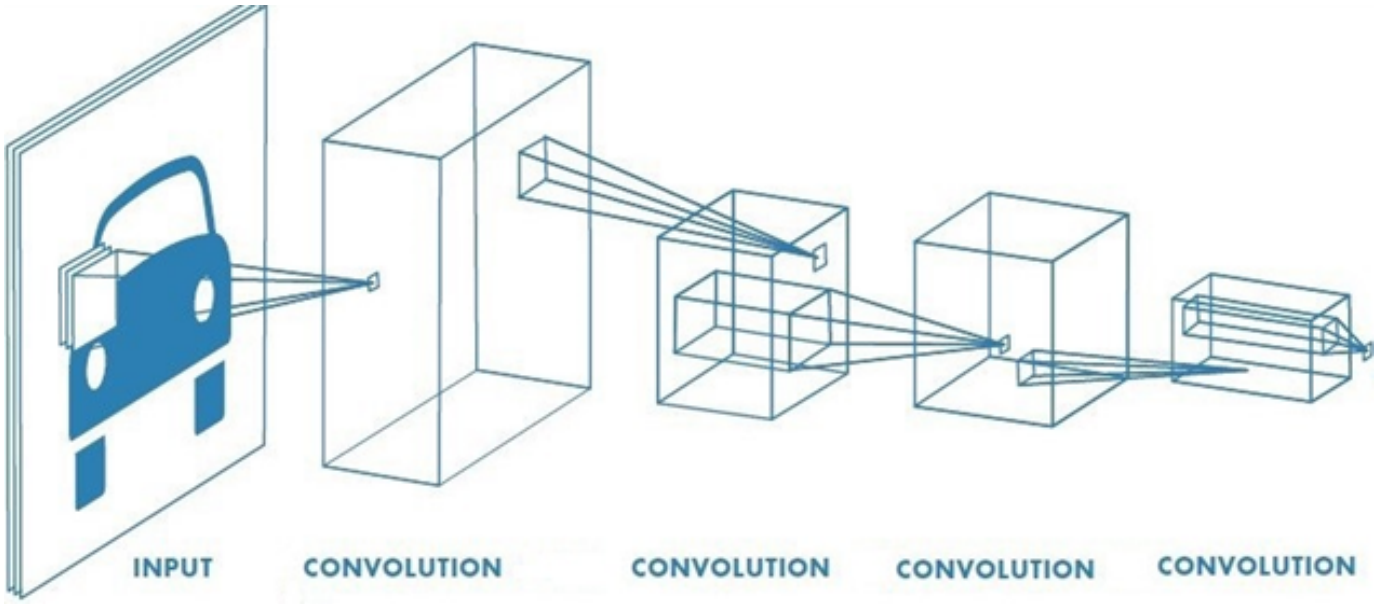


Figure 3.18 - Convolutional Neural Network

2. **Feeding these features into the Generative Adversarial Network:**

Now after the Convolutional Neural Network layers have extracted features from the images, these extracted features are then input into the Generative Adversarial Network. The Generative Adversarial Network, consists of two parts -

1. The Generator
2. The Discriminator

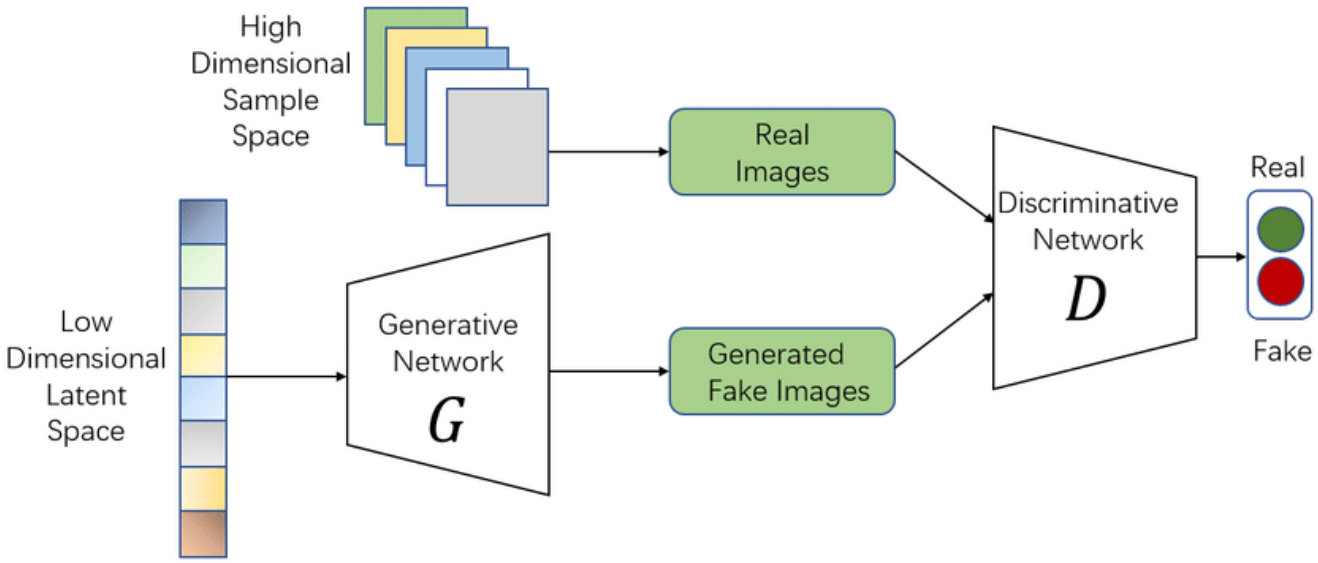


Figure 3.19 - Generative Adversarial Network (GAN)

3. **The Generator:**

Generator is the model that is used to generate new example images from scratch from the given input image.

What Generator actually does is that it captures the data distribution of the image and creates new example images and the Discriminator estimates the probability that the example image generated from the Generator came from the training data rather than the Generator. The training procedure for the Generator is to maximize the probability of the Discriminator making a mistake and thinking that the image produced by the Generator is from the training set instead.

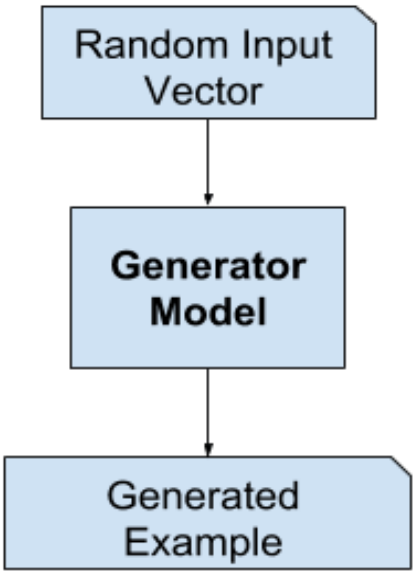


Figure 3.20- Generator Model

4. **The Discriminator:**

The discriminator is provided both with an input image which is real and the image generated by the Generator. Its work is to classify these images as either real (from the input/the training set) or fake (generated by the generator).

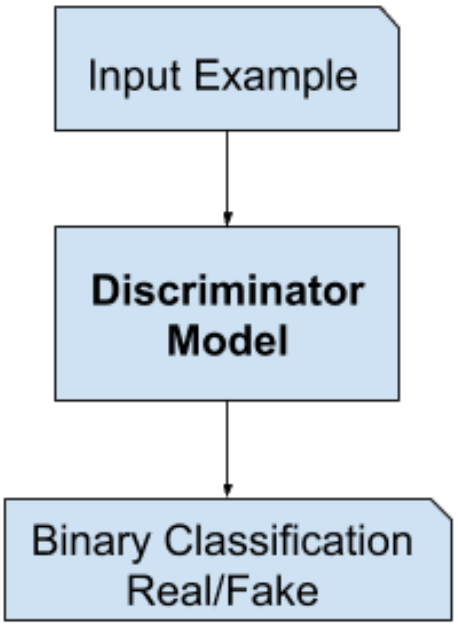


Figure 3.21 - Discriminator Model

5. Creating Compressed Images:

After the Generator and the 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.

Figure 3.22 - Image Before and After Compression

Chapter 4: Modeling and Implementation Details

4.1 Design diagram

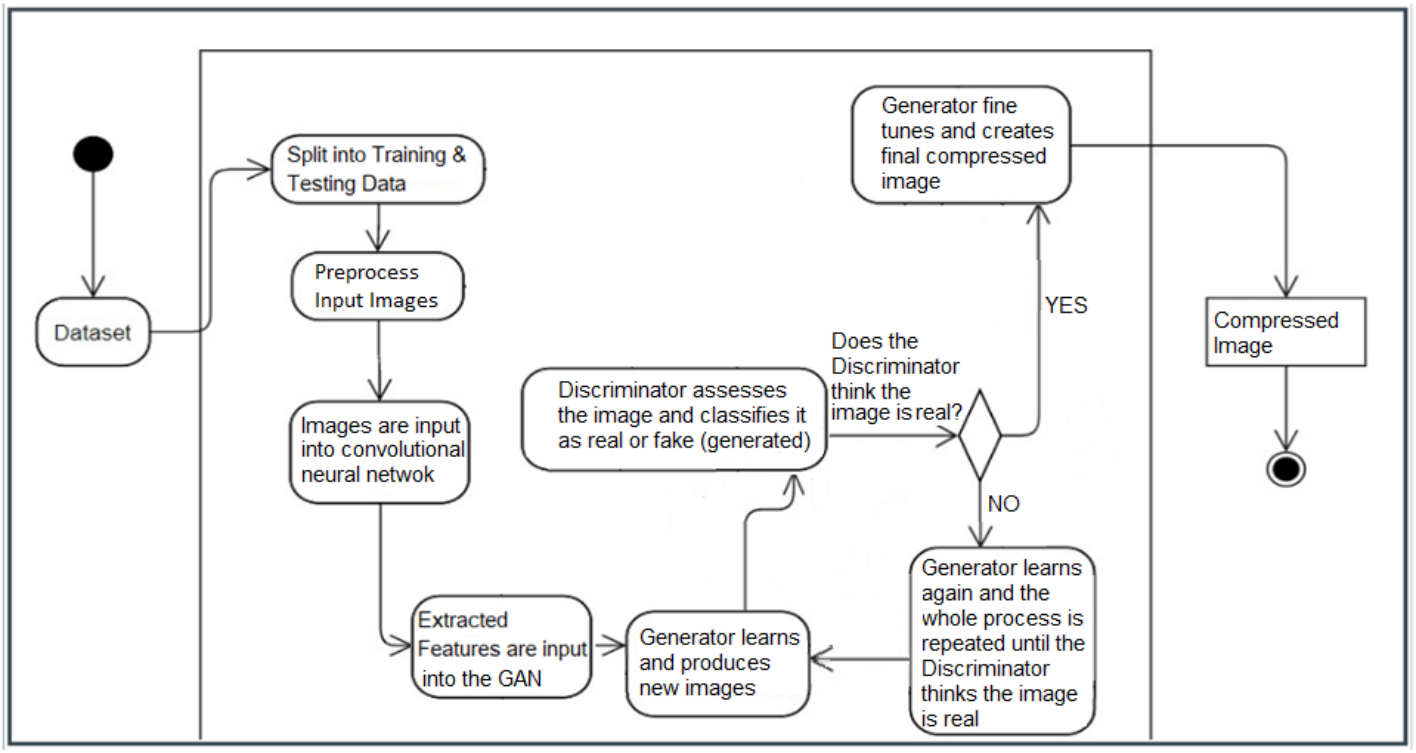


Figure 4.1 - Design Diagram of our Model

4.2 Implementation Details

These are the steps we did for implementing our proposed solution:

1. **Collecting the datasets:**

   We collected the CLIC2020 and Kodak and DIV2K datasets. We sorted them in order and all the images were checked if they were not corrupt. We also collected a subset of the ImageNet Dataset which contains around 9650 images of wide variety, which we use for training and testing our model.

   We divided the collected dataset into training and testing data in the ratio 80% to 20%. This is important to be done, so that the model can learn and get trained on the input images.

2. **Preprocess the input image files:**

   We wrote a python program and used the OpenCV python library for this. This is done, so that the Convolutional Neural Network can learn from these images.

Figure 4.2 - OpenCV Library

3. Detect and Extract Features:

We wrote a Machine learning model made up of Convolutional layers and Maxpool2D layers in between them.

The Convolutional Neural Network extracts spatial features from the images and trains on them, so that it can better predict which information in the image is to be selected for compression and which not to be selected.

```
20:18:33] Epoch: 10/300 Loss: 0.6691/0.7211/1.3902
20:18:35] Epoch: 11/300 Loss: 0.6643/0.6916/1.3558
20:18:38] Epoch: 12/300 Loss: 0.6523/0.7220/1.3743
20:18:40] Epoch: 13/300 Loss: 0.6333/0.6979/1.3312
20:18:43] Epoch: 14/300 Loss: 0.6321/0.6748/1.3069
20:18:45] Epoch: 15/300 Loss: 0.6221/0.6648/1.2869
20:18:48] Epoch: 16/300 Loss: 0.6118/0.6553/1.2671
20:18:50] Epoch: 17/300 Loss: 0.6026/0.6504/1.2530
20:18:52] Epoch: 18/300 Loss: 0.6001/0.6685/1.2686
20:18:55] Epoch: 19/300 Loss: 0.5809/0.6502/1.2311
```

```
20:19:00] Epoch: 21/300 Loss: 0.5741/0.6385/1.2127
20:19:02] Epoch: 22/300 Loss: 0.5608/0.6241/1.1849
20:19:05] Epoch: 23/300 Loss: 0.5562/0.6175/1.1737
20:19:07] Epoch: 24/300 Loss: 0.5497/0.6143/1.1640
20:19:09] Epoch: 25/300 Loss: 0.5456/0.5866/1.1322
20:19:12] Epoch: 26/300 Loss: 0.5324/0.6024/1.1348
20:19:14] Epoch: 27/300 Loss: 0.5167/0.6271/1.1438
20:19:17] Epoch: 28/300 Loss: 0.5318/0.5948/1.1265
20:19:19] Epoch: 29/300 Loss: 0.5243/0.6057/1.1300
20:19:21] Epoch: 30/300 Loss: 0.5178/0.5773/1.0951
20:19:24] Epoch: 31/300 Loss: 0.5020/0.5733/1.0753
20:19:26] Epoch: 32/300 Loss: 0.4968/0.5636/1.0604
20:19:28] Epoch: 33/300 Loss: 0.5022/0.5620/1.0641
20:19:31] Epoch: 34/300 Loss: 0.4909/0.5665/1.0574
20:19:33] Epoch: 35/300 Loss: 0.4924/0.5656/1.0579
```

Figure 4.3 - Extracting features from the images

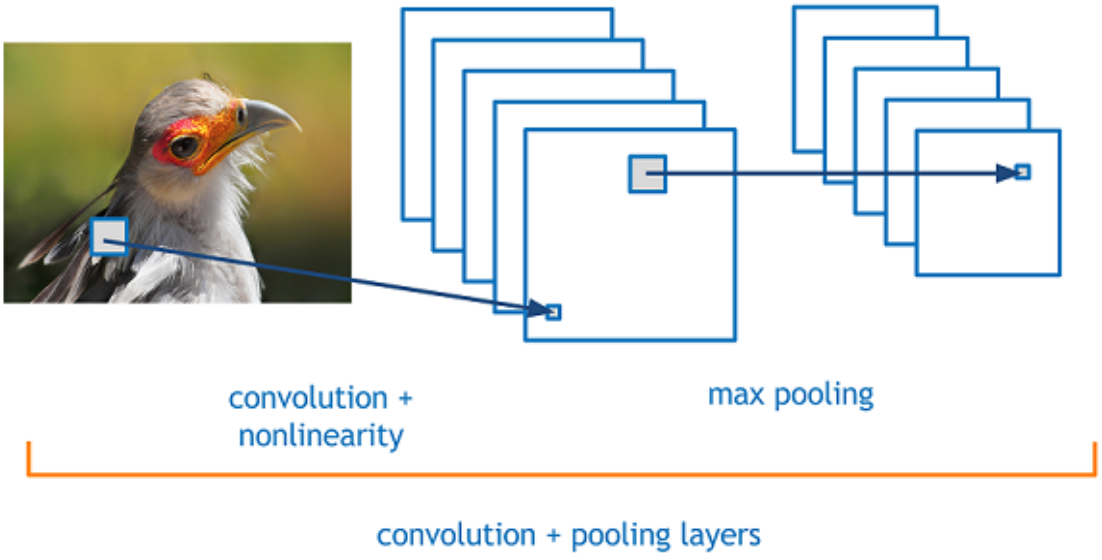


Figure 4.4 - Convolutional Neural Network

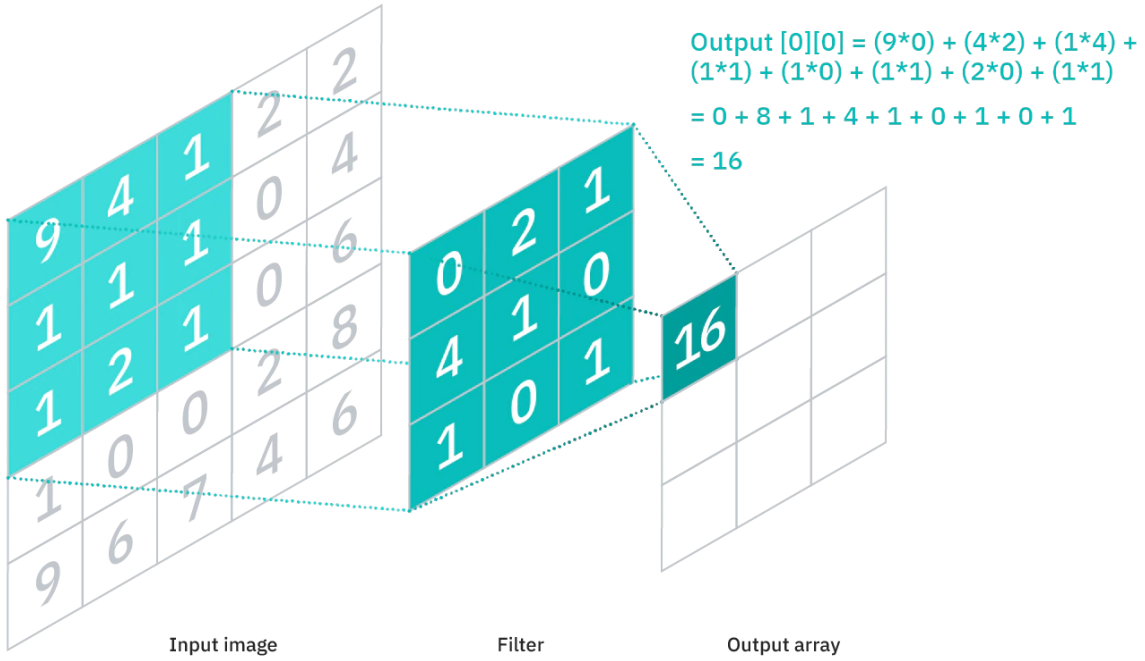


Figure 4.5 - Convolutional Neural Network learning features of the input image

4. Reducing Overfitting:

We noticed that the model was overfitting, so we added dropout layers with a chance of 0.5 in between the convolutional layers. This helps reduce overfitting. The output of these layers gives us the features our model has extracted from the images

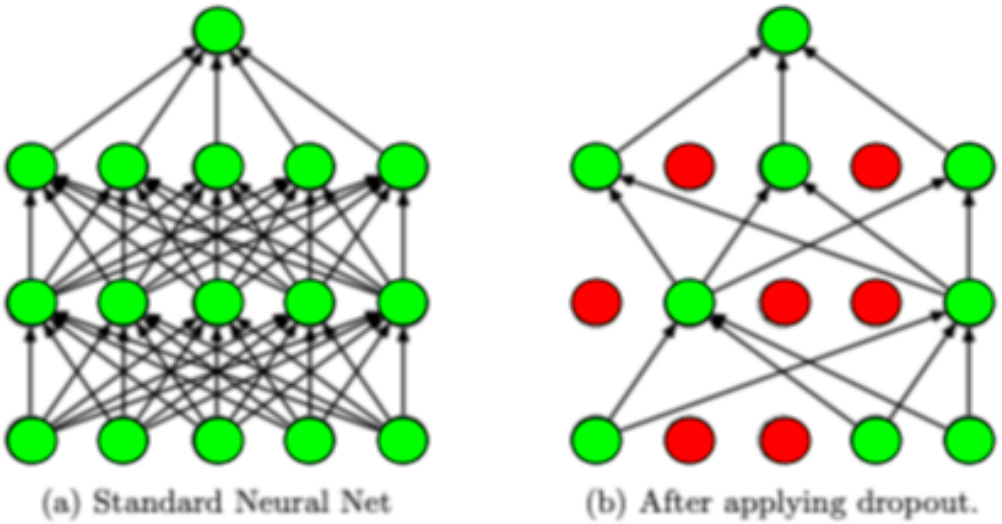


Figure 4.6 - Applying Dropout Layers to reduce overfitting

5. Feeding these extracted features into GAN:

After extracting Spatial Features via Convolutional layers, these extracted features are then input into the (GAN) Generative Adversarial Network.

The Generative Adversarial Network, consists of two parts -

The Generator and

The Discriminator

We have used GANs (Generative Adversarial Networks) because Generative Adversarial Networks don't break down with high-dimensional data. They work well with unsupervised data.

Currently, out of all other options, they produce the sharpest images. Another advantage is that both networks, Generator and the Discriminator in GANs can be trained with only Backpropagation.

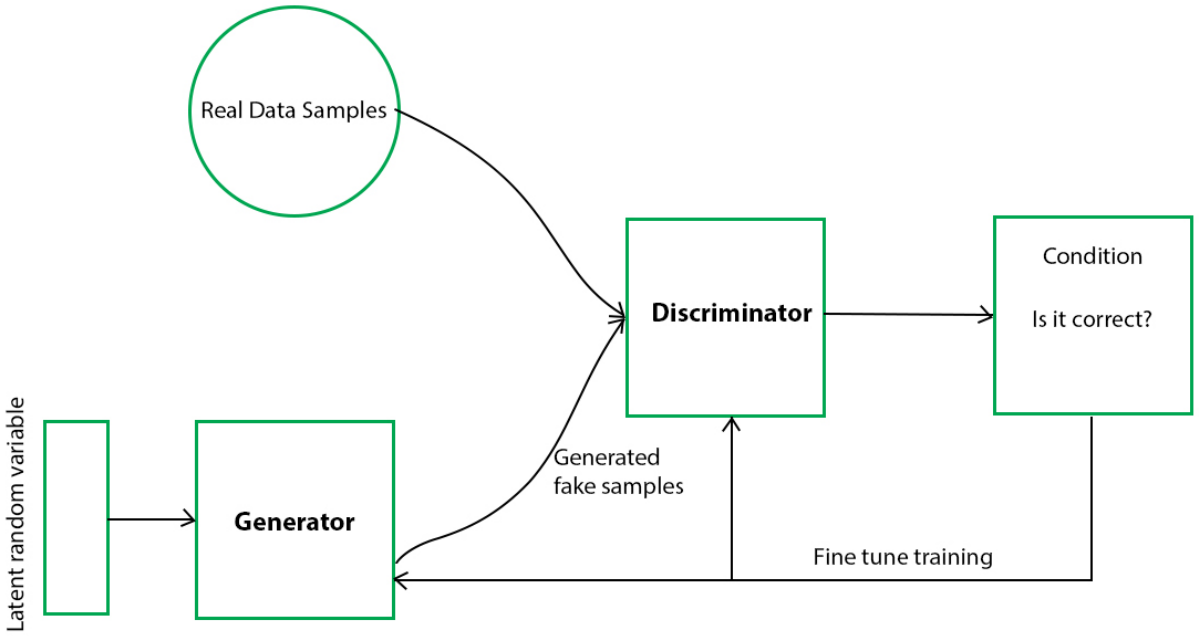


Figure 4.7 - Structure of Generative Adversarial Network (GAN)

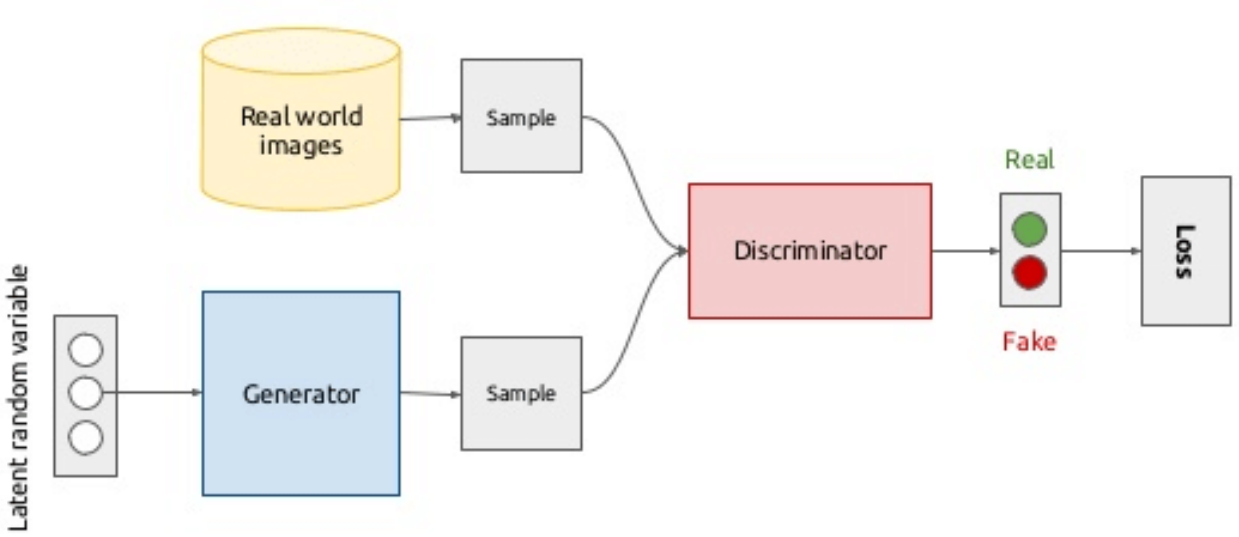


Figure 4.8 - Generative Adversarial Network (GAN) learning and creating new images and assessing created images

6. **Generator learns and produces new images:**

The Generator starts learning from the features it is given by Convolutional Neural Network and tries to generate new example images from scratch. The Generator tries to maximize the probability of the Discriminator making a mistake and thinking that the image produced by the Generator is from the training set instead. This is how the Generator gets trained.

In the end, the Generator is unknowingly gets trained and gets forced to create images which are very similar to the original images.

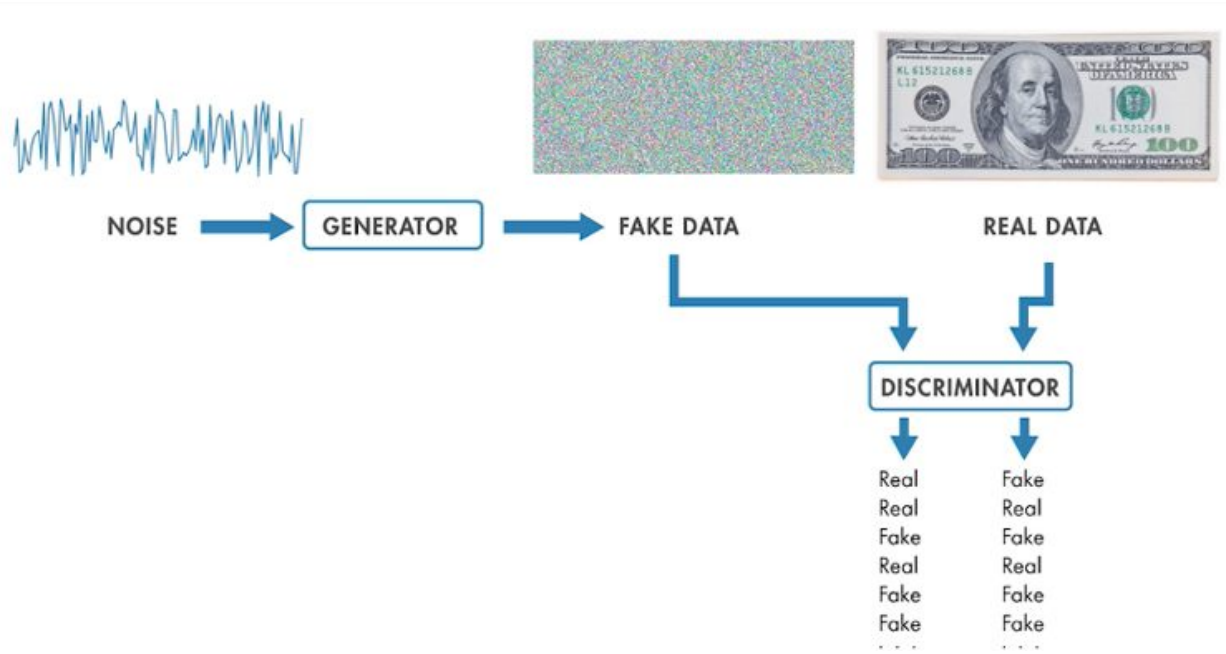


Figure 4.9 - Generative Adversarial Network (GAN) assessing images created by the Generator

7. **Discriminator starts assessing the images produced by the Generator:**

What Generator actually does is that it captures the data distribution of the image and creates new example images and the Discriminator estimates the probability that the example image generated from the Generator came from the training data rather than the Generator.

The discriminator is provided both with an input image which is real and the image generated by the Generator.

Discriminator classifies whether the image generated by the Generator is real or fake, and returns its assessment to the Generator, in turn teaching the generator to generate images which fool the discriminator.

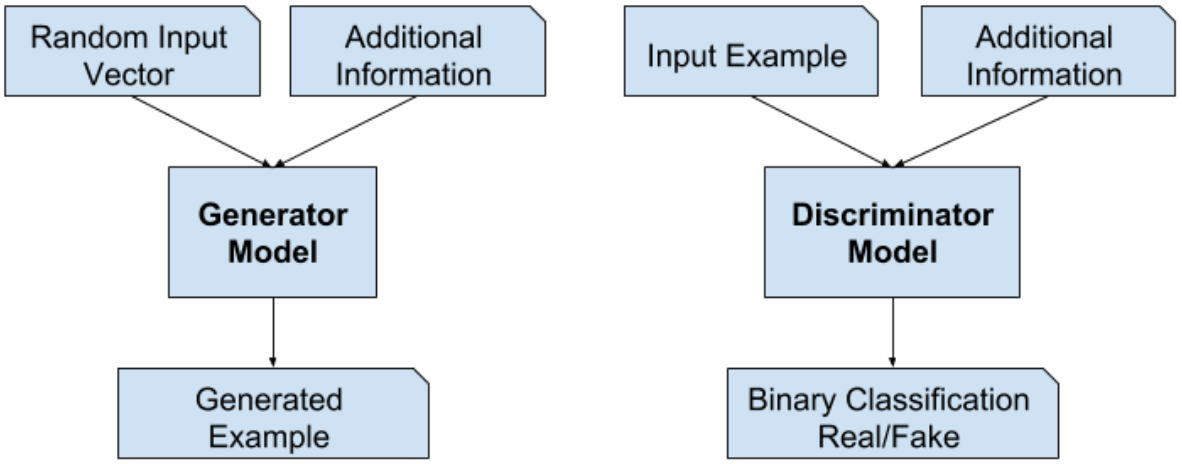


Figure 4.10 - Generator creating new images and Discriminator assessing them

8. Training the model:

We trained the model for 20,000 iterations for 36 hours. We made changes to the model multiple times and trained the model again to increase its accuracy. 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.

After this the model became robust and was able to find which information in the image is important and needs to be included in the compressed image. Our model was able to create visually pleasing compressed images which were much smaller in size than the original image.

```
                                    nishit@nishit-ubuntu: ~/Downloads/D
20:59:50] Epoch: 286/300 Loss: 0.1905/0.2707/0.4612
20:59:51] Epoch: 287/300 Loss: 0.1690/0.2677/0.4367
20:59:52] Epoch: 288/300 Loss: 0.1680/0.2633/0.4313
20:59:52] Epoch: 289/300 Loss: 0.1717/0.2937/0.4654
20:59:53] Epoch: 290/300 Loss: 0.1757/0.2930/0.4688
20:59:54] Epoch: 291/300 Loss: 0.1601/0.2885/0.4486
20:59:55] Epoch: 292/300 Loss: 0.1754/0.2902/0.4656
20:59:56] Epoch: 293/300 Loss: 0.1806/0.2610/0.4417
20:59:57] Epoch: 294/300 Loss: 0.1637/0.2765/0.4402
20:59:58] Epoch: 295/300 Loss: 0.1562/0.2646/0.4208
20:59:58] Epoch: 296/300 Loss: 0.1679/0.3195/0.4874
20:59:59] Epoch: 297/300 Loss: 0.1641/0.2772/0.4414
21:00:00] Epoch: 298/300 Loss: 0.1668/0.2926/0.4594
21:00:01] Epoch: 299/300 Loss: 0.1558/0.2874/0.4432
```

Figure 4.11 - Training our model on all four datasets

9. Output Compressed Images:

After the Generator and the 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.

In Generative Adversarial Network, 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.

Figure 4.12 - Output Compressed image given by our model

10. 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 GPU - GTX 1050Ti Max Q had 4GB of VRAM. Compared to server GPUs which have upto 24GB of GPU VRAM, 4GB was much less. 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.

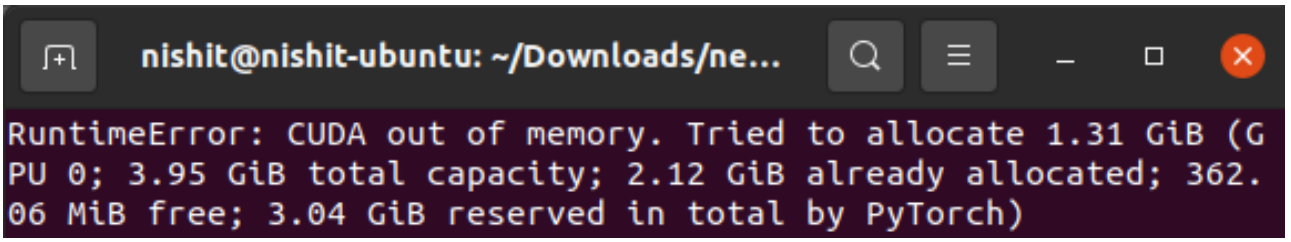


Figure 4.13 - Our model running out of memory while compressing images, before we optimized it

Now, after this, the model became much faster and now it is able to compress images upto 1500kB (1.5 MB) which is 3 times more as compared to before and upto 1000x1000 resolution which is 6 times more resolution as compared to before.

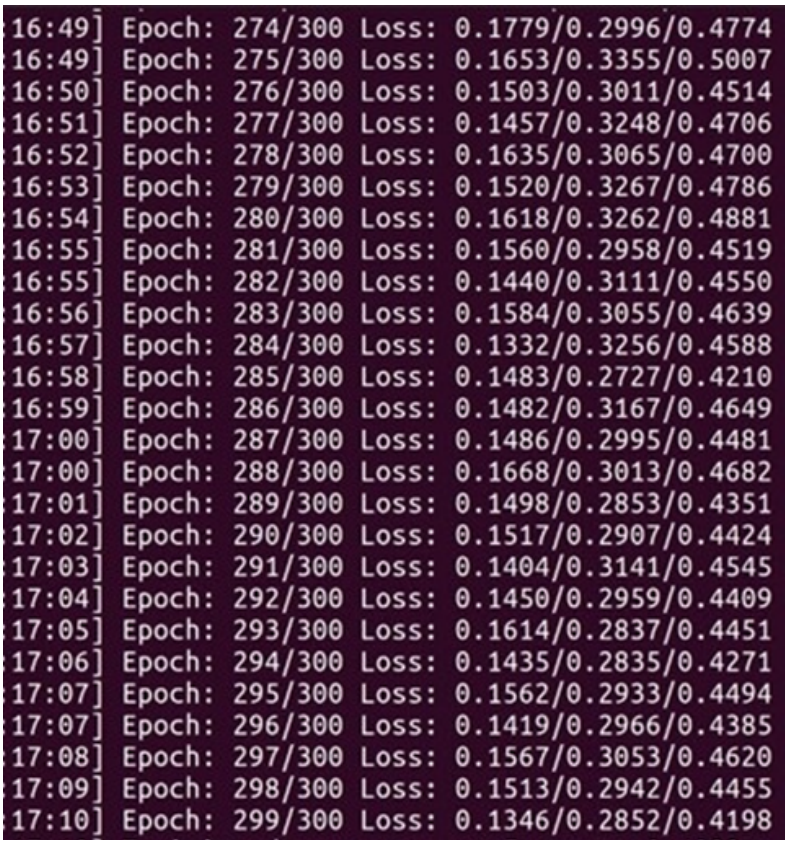
Chapter 5: Testing

We originally split our dataset containing Kodak dataset, as well as the CLIC2020 dataset and DIV2K dataset along with a subset of the Imagenet Dataset for testing and training purposes, so now we evaluated the model by testing it against the testing dataset splits of the original datasets.

5.1 Testing details

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.

Finally, we achieved a PSNR Ratio of 36 on the datasets which is quite higher as compared to previous techniques in the field of Image Compression.



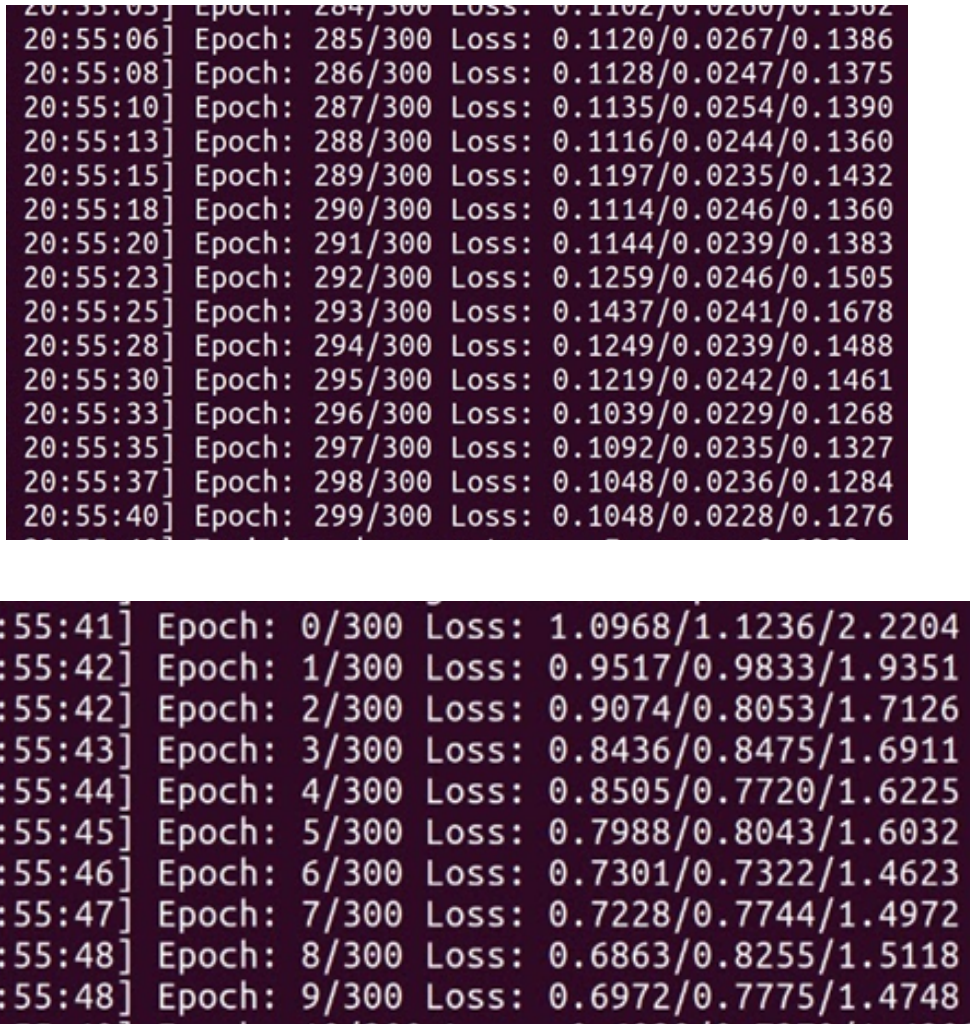


Figure 5.1 - Testing our model on our datasets

5.2 Issues and limitations of the solution :

Issues faced by us during the implementation were-

1. Some of the images in the datasets were too hazy and of bad quality, which reduced the accuracy of the model.
2. As we use GANs, it requires a lot of resources and GPU Memory. Thus many times, while training the model, we ran out of GPU memory and had to start training the model from the start.

Chapter 6 : Observation and Results

6.1 Result and Findings

1. 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.

   Here is the output size of images as compared to input size of 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.

   These are the size of images before and after compression-

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

2. Finally, we achieved a PSNR Ratio of 36 on the datasets which is quite higher as compared to previous techniques.

$$MSE = \frac{\sum_{M,N} [I_1(m,n) - I_2(m,n)]^2}{M * N}$$

$$PSNR = 10 \log_{10}\left(\frac{R^2}{MSE}\right)$$

   Here is our PSNR Ratio as compared to the work of some previous authors on the topic of Image compression on our dataset-

Model	PSNR (Peak Signal to Noise Ratio)
JPEG	31
BPG	34.5
Mentzer et al.	33
Toderici et al.	26
Ours	36

Table 6.2 - PSNR ratio of our model compared to other image compression models and algorithms

Before Compression (1.0 MB)

After Compression (30.5 KB)

Before Compression (1.1 MB)

After Compression (34.0 KB)

Before Compression (1.1 MB)

After Compression (27.8 KB)

Figure 6.1 - Comparison of images before and after compression by our model

3. Earlier our model could only compress images upto 500kB and upto 400x400 resolution.

   We optimized our model and made it lightweight compared to other Image Compression models by reducing a few layers from the model.

   Now, after this, the model became much faster and now it is able to compress images upto 1500kB (1.5 MB) which is 3 times more as compared to before and upto 1000x1000 resolution which is 6 times more resolution as compared to before.

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 6.3 - Our model before and after optimization

6.2 Gantt Chart

Try to reduce the GPU VRAM required for compression	Try to compress images of larger size and dimension than currently our model is able to do

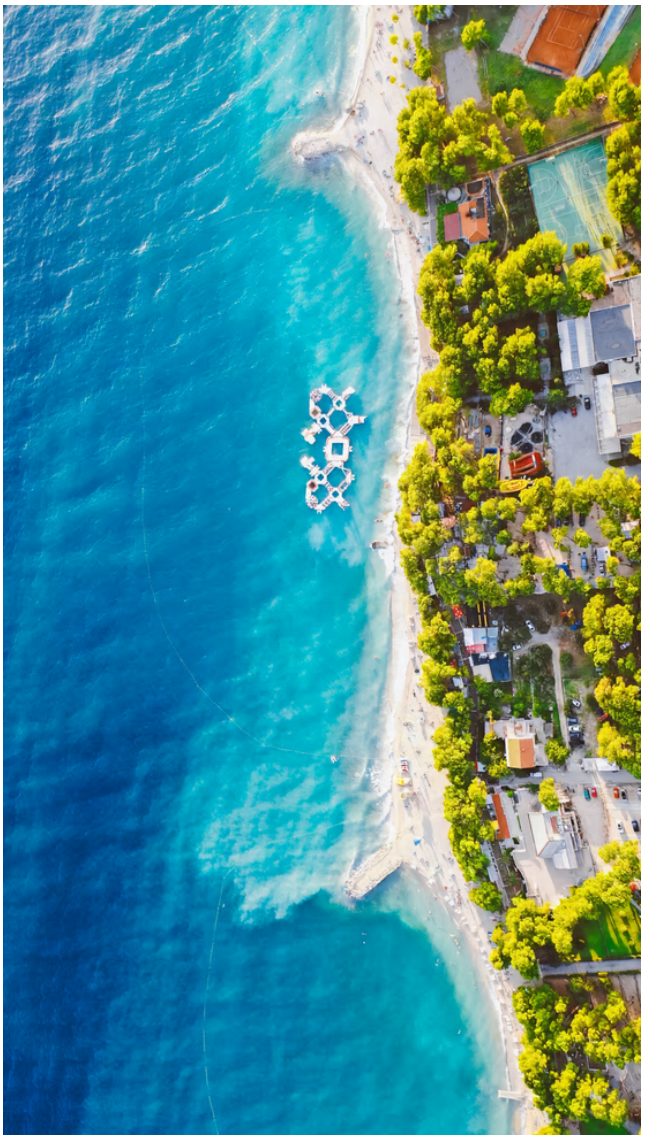
June July August

Table 6.4 - Gantt Chart for future plan

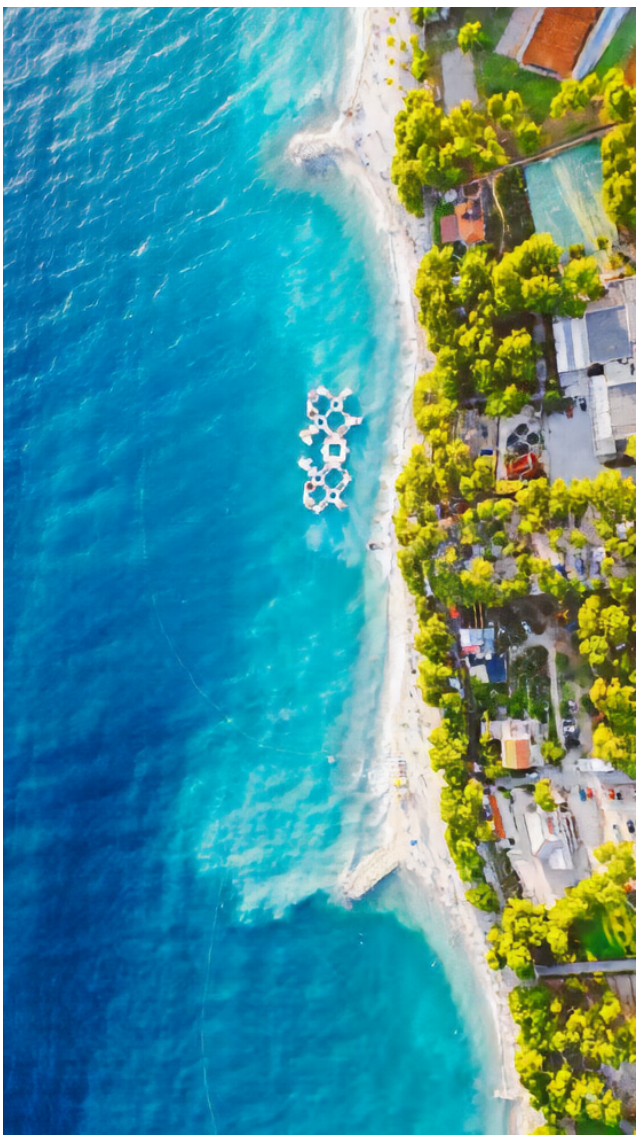
6.3 Future Plan and Conclusion

In this report, we have proposed our idea to create a framework for image compression. Unlike existing methods, we have used Generative Adversarial Networks, which help reduce the size of output image by quite a bit as well as help in maintaining image quality. 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)



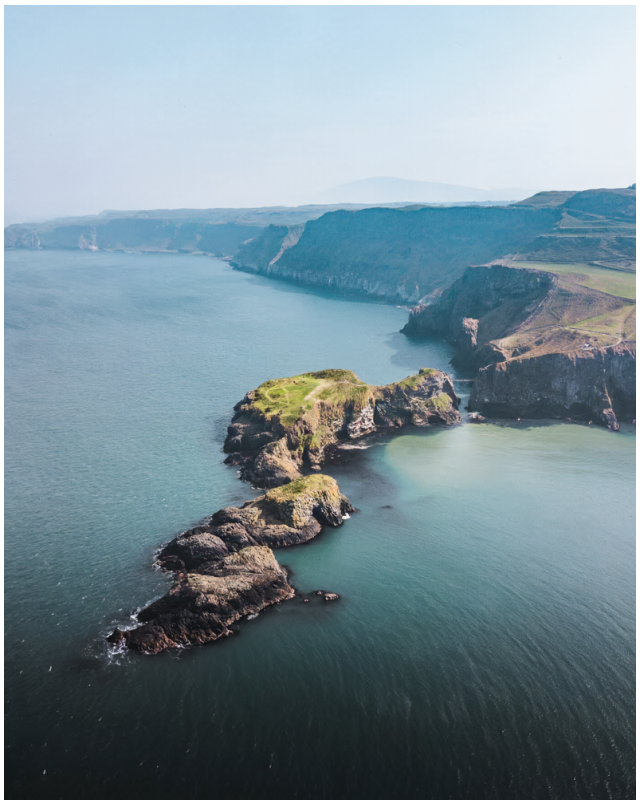
Before Compression (980.8 KB)



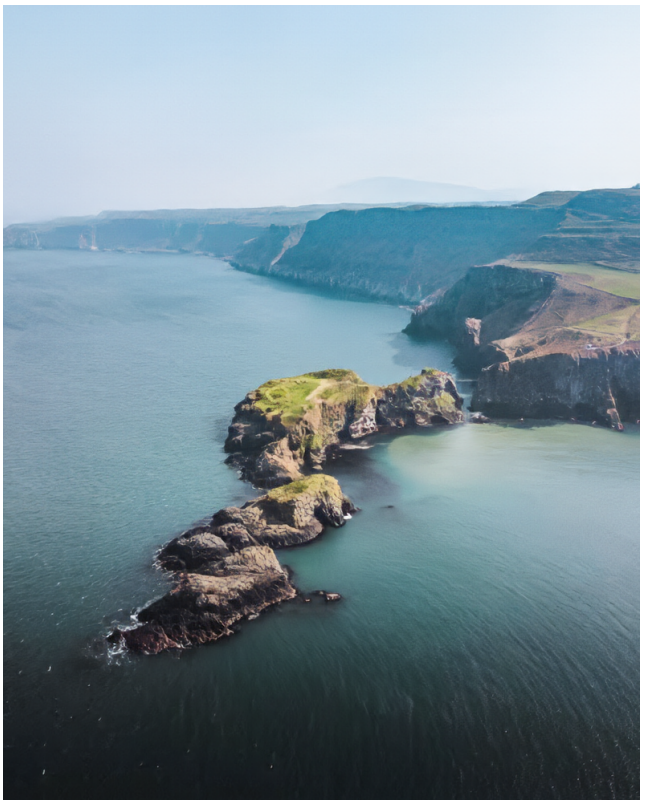
After Compression (29.9 KB)

Before Compression (1.0 MB)

After Compression (25.0 KB)



Before Compression (1.1 MB)



After Compression (23.1 KB)

Before Compression (870KB)

After Compression (20.5 KB)

Figure 6.2 - Comparison of few more images before and after compression by our model

References-

[1] Fatih Kamisli, “End-to-End Learned Block-Based Image Compression with Block-Level Masked Convolutions and Asymptotic Closed Loop Training”, 2022

[2] Xiaosu Zhu, Jingkuan Song, Lianli Gao, Feng Zheng, Heng Tao Shen , “Unified Multivariate Gaussian Mixture for Efficient Neural Image Compression”, 2022.

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

[4] Yoshitomo Matsubara, Ruihan Yang, Marco Levorato, Stephan Mandt, “Supervised Compression for Split Computing”, WACV 2022

[5] Yichen Qian, Ming Lin, Xiuyu Sun, Zhiyu Tan, Rong Jin, “Entroformer: A Transformer-based Entropy Model for Learned Image Compression”, 2022

[6] Mingtian Zhang, James Townsend, Ning Kang, David Barber,“Parallel Neural Local Lossless Compression”, 2022

[7] Mu Li, Kede Ma, Jinxing Li, David Zhang ,“Pseudocylindrical Convolutions for Learned Omnidirectional Image Compression”, Picture Coding Symposium (PCS), Numberg, Germany, 2016

[8] Yuanchao Bai, Xu Yang, Xianming Liu, Junjun Jiang, YaoWei Wang, Xiangyang Ji, Wen Gao, “Towards End-to-End Image Compression and Analysis with Transformers”, AAAI 2022

[9] Jae-Han Lee, Seungmin Jeon, Kwang Pyo Choi, Youngo Park, Chang-Su Kim,“DPICT: Deep Progressive Image Compression Using Trit-Planes”, 2021

[10] Benoit Brummer, Christophe De Vleeschouwer,“End-to-end optimized image compression with competition of prior distributions”, CVPRW 2021

[11] Mercy G. Amankwah, Daan Camps, E. Wes Bethel, Roel Van Beeumen, Talita Perciano,“Quantum pixel representations and compression for N-dimensional images”, 2021

[12] Jiaxi Jiang, Kai Zhang, Radu Timofte, “Towards Flexible Blind JPEG Artifacts Removal”, 2021

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

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

[15] Yueqi Xie, Ka Leong Cheng, Qifeng Chen ,“Enhanced Invertible Encoding for Learned Image Compression”, ACMMM 2021

9918103133_Image Compression using Deep Learning

ORIGINALITY REPORT

16%	13%	6%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	arxiv.org Internet Source	2%
2	Submitted to Indian Institute of Technology Guwahati Student Paper	2%
3	Aakriti Aggarwal, Siddhant Wadhwa, Pallav Gupta, Nishit Anand, Rashmi Kushwah. "IsSwap: Deep Fake Detection", 2021 7th International Conference on Signal Processing and Communication (ICSC), 2021 Publication	1%
4	medium.com Internet Source	1%
5	www.coursehero.com Internet Source	1%
6	mavenlin.github.io Internet Source	1%
7	de.slideshare.net Internet Source	1%

8	ui.adsabs.harvard.edu Internet Source	1 %
9	Submitted to City University Student Paper	<1 %
10	cloud.tencent.com Internet Source	<1 %
11	M.H. Hayes. "Image coding with polynomial transforms", Conference Record of The Thirtieth Asilomar Conference on Signals Systems and Computers ACSSC-96, 1997 Publication	<1 %
12	Submitted to Higher Education Commission Pakistan Student Paper	<1 %
13	Ye Wang, Yueru Chen, Jongmoo Choi, C.-C. Jay Kuo. "Towards Visible and Thermal Drone Monitoring with Convolutional Neural Networks", APSIPA Transactions on Signal and Information Processing, 2019 Publication	<1 %
14	Submitted to Monash University Student Paper	<1 %
15	developer.skatelescope.org Internet Source	<1 %
16	openaccess.thecvf.com Internet Source	<1 %

17	www.ibm.com Internet Source	<1%
18	Submitted to University of East London Student Paper	<1%
19	fugumt.com Internet Source	<1%
20	Submitted to Ghana Technology University College Student Paper	<1%
21	Misaki Kanai, Ren Togo, Takahiro Ogawa, Miki Haseyama. "Synthetic Image Generation for Gastritis Detection Based on Auxiliary Classifier Generative Adversarial Network", 2019 IEEE International Symposium on Circuits and Systems (ISCAS), 2019 Publication	<1%
22	www.sigmm.org Internet Source	<1%
23	export.arxiv.org Internet Source	<1%
24	Submitted to University of Nottingham Student Paper	<1%
25	www.hindawi.com Internet Source	<1%
26	www.essays.se Internet Source	

		<1%
27	www.semanticscholar.org Internet Source	<1%
28	bookpedia.co Internet Source	<1%
29	paperreading.club Internet Source	<1%

Exclude quotes	On	Exclude matches	< 14 words
Exclude bibliography	On		

NISHIT ANAND

nishitanand01@gmail.com | **+91 9899218301** | https://www.linkedin.com/in/nishitanand | https://www.github.com/nishitanand

EDUCATION

Education level	Institution Name	Grade
B.Tech (Computer Science)	Jaypee Institute of Information Technology (JIIT),Noida	8.7 CGPA
Class XII	Apeejay School Pitampura, New Delhi (CBSE Board)	93 %
Class X	Apeejay School Pitampura, New Delhi (CBSE Board)	9.6 CGPA

PROJECTS

1. **SaveLives – A Real-Time Threat Detection System**
   - Detects **guns, knives and faces of registered people** in video **footage of CCTV cameras** so that authorities can be notified when a **gun or knife is detected** with a **terrorist** or **thief** and intervene before the crime happens
   - Achieved **Highest Mean Average Precision**(mAP) of **90.35%** and Highest Average IOU of 73.68% on validation dataset
2. **LightningTransfer – A Huffman Coding Compression Project**
   - Converts any type of file like **JPEGS, MP3, PDFs, MP4 to text file**
   - Then compresses this text file using Huffman Coding. File size **reduced by 25%**,so that they can be transferred blazingly fast and then uncompressed on the other end
3. **DeMystifying Online Buying - Real Time Price Comparison**
   - Uses **Web Scraping** to get **Real time Price** of a product on **Amazon** and **Snapdeal**
   - Uses **Flask** For **Backend** as well as Routing
   - Displays the results on a web page using **HTML** and **Django**

PUBLICATIONS

SaveLives – A Real-Time Threat Detection System (**Accepted at ICI 2022** - International Conference on Informatics 2022)
- Created an object detection framework which detects guns and knives in video footage and alerts the police instantaneously to stop a crime from occuring.

IsSwap: Deep Fake Detection (**Accepted at ICSC 2021** - International Conference on Signal Processing and Communication 2021)
- Created a Custom Deep Learning pipeline using **ResNeXt-50** and **LSTM** which is used to predict whether a given video is real or fake.

EXPERIENCE

1. **Software Development Engineer Intern -** Pepcoding Pvt. Ltd. (June 2021 - August 2021)
   - Worked on the **Backend** of the **Doubt Support Feature**
   - Wrote code for **connecting sockets** in **Python** on the Pepcoding website using **Django** and **Flask**
2. **Software Engineering Virtual Experience** - JPMorgan Chase And Co. (May 2020 - May 2020)
   - Worked on **Data analysis** and **Data Visualization** along with designing assessment modules and test modules.
3. **Technical Coordinator** - JIIT Open Source Developers Circle (July 2019 – September 2021)
   - **Taught 100+ Student**s how to use **Git and Github**
   - **Conducted workshops** about **GSoC (Google Summer of Code)** and how to contribute to **Open Source**
   - Organized various workshops to help college students learn about **Linux, Open source** and helping them work on **real world projects**

AWARDS

- **National Level** Coding Competition - **TechGig Code Gladiators Finalist** | **Rank 464 out of 252365 (2.5 lakh)** participants
- **International Coding Competition** - Received Certificate of Achievement - **Honorable Mention in ACM-ICPC 2019-20**
- Secured **8th position out of 881** participants in Algofuzz 2019 held at JIIT-128 Noida
- Secured **14th Position at Project Euler** Competition held at **IIIT Delhi** (Esya 2019)
- Secured **16th Position at Algoflux 2020** at JIIT-128 Noida
- Received **Certificate of Completion of KWOC 2019 (Kharagpur Winter of Code)** organized by (KOSS) Kharagpur Open Source Society which is the **Open Source Society of IIT Kharagpur**

SKILLS

Proficient:

C++
Python
Data Structures
Competitive Programming

Intermediate:

Algorithms
Deep Learning
Machine Learning
Computer Vision

Shreya Agarwal

shreya.sv495@gmail.com | +91-9828366641
KPS Residency
Kaushallya More,Kharagpur-721301
West Bengal

ABOUT

Raising engineer in computer science and technology

I am a hard-working and desiring person, aiming to become a successful engineer soon. Striving to fill in great experiences in this resume soon.

EDUCATION

Btech in Computer Science engineering
Jaypee Institute of information technology,Noida
2018-2022
9.0 grade point average

Emmanuel Mission School
2015-2017
Kota, Rajasthan, India

St.Agnes School
Kharagpur,West Bengal, India

10th 85 percent | 12th 82 percent

SKILLS

PROGRAMMING

Data structures • Algorithms • Problem Solving

LANGUAGES

C • C++ • SQL • HTML • CSS • JAVASCRIPT
Familiar:
• React • Python

COURSEWORK

UNDERGRADUATE

- Data Structures
- Algorithms and Problem Solving
- Operating System
- Database Management System

SELF-LEARNING

Front end framework with react
Application development with flutter

LINKS

Github:// **Shreya-495**
LinkedIn:// **shreya-sv495**

PROJECTS

CODESHARING APPLICATION

A real time code sharing application using NodeJs, ExpressJs and MongoDb as a database. It allows users to write live codes and also have an integrated chat facility.

COMMUNITY OF DOCTORS

A react based web application serving as a platform for the doctors and medical professionals from around the globe to post journals, share findings and discuss the latest happenings in the field of medicinal sciences.

HOTEL MANAGEMENT SYSTEM

- A single paged application build with the help of html,css,javascript and php.
- Allows users to check in and out into our hotel.
- Proper sign in,sign up, menu, available rooms and feedback pages.

SNAKE AND FOOD GAME

The classic snake and the food game written and compiled in c.

ONE-STOP-RIDE

A flutter based application to compare cab fares from differnt service providers and return the best fit fare to the user.

CERTIFICATIONS

COURSERA AND GCP COURSES

- Cloud Development - Qwiklabs
- Websites and Web Applications - Qwiklabs
- Exploring APIs - Qwiklabs
- Deploying Applications - Qwiklabs
- Baseline: Deploy Develop - Qwiklabs
- Front-End Web UI Frameworks and Tools: Bootstrap 4 (with Honors) - Coursera
- Front-End Web Development with React (with Honors) - Coursera

Aditi Dixit

Final Year (B.Tech)
Computer Science & Engineering

Kanpur
Uttar Pradesh,India
Mob.: +91-6387307351
Email.:aditidixit789@gmail.com

Links

HackerRank:// **AditDixit**
CodeChef:// **AditiDixit**

Skills

LANGUAGES
C/C++, HTML, CSS, javascript

OTHERS
SQL
Data Structures
Algorithms And Problem Solving

Coursework

Data Structures
Algorithms And Problem Solving
Database Systems And Web
Operating System
Software Engineering

Hobbies/Interests

Reading
Journaling
Gardening
Music

Education

Graduation Details
August 2018-Present (Expected: May 2022)
B.TECH IN CSE
Jaypee Institute Of Information Technology
Noida, Uttar Pradesh
CGPA:8.1 till sixth sem

Intermediate
2016-2017
PTN Inter College
Kanpur,Uttar Pradesh
Percentage: 89

High School
2014-2015
Mercy Memorial School
Kanpur,Uttar Pradesh
Percentage: 95

Achievements

5-star badge | Problem Solving | HackerRank
3-star | CodeChef

Projects

College Website
This project is on Web Development and Architecture and Database systems associated with it.
• It is a college website that takes into account factors like students login, parents login, faculty login, contact information etc.
• It is build with the help of HTML, CSS, javascript, php and SQL.

Snake And Food Game
The classic snake and food game written and compiled in C, which provides the following functionalities:
• Snake can move in a given direction and when it eats the food, the length of snake increases.
• When snake crosses itself, the game is over.
• Food will be generated at a given interval.