



***DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING,
SHARDA SCHOOL OF ENGINEERING AND TECHNOLOGY,
SHARDA UNIVERSITY, GREATER NOIDA***

Image Processing using Deep Learning Techniques: Super Resolution

***A project report submitted
in partial fulfillment of the requirements for the award of the degree of
Bachelor of Technology in Computer Science and Engineering***

by

ASHIMA JAIN (2019003789)

ARUP SARKAR (2019000503)

Supervised by:

Mr. Shobhit Tyagi, Assistant Professor

Dr. Arun Prakash Agrawal, Professor

May, 2023

CERTIFICATE

This is to certify that the report entitled “**Image processing using Deep Learning Techniques: Super Resolution**” submitted by “**ASHIMA JAIN (2019003789) and ARUP SARKAR (2019000503)**” to Sharda University, towards the fulfillment of requirements for the award of the degree of “**Bachelor of Technology in Computer Science and Engineering**” is record of bonafide final year Project work carried out by them in the “**Department of Computer Science & Engineering, Sharda School of Engineering and Technology, Sharda University**”. The results/findings contained in this project have not been submitted in part or full to any other University/Institute for award of any other Degree/Diploma.

Signature of the Guide

Name: Mr. Shobhit Tyagi

Designation: Assistant Professor (CSE)

Signature of Head of Department

Name: Prof. (Dr.) Nitin Rakesh

Place: Sharda University

Date:

Signature of External Examiner

Date:

ACKNOWLEDGEMENT

A major project is a golden opportunity for learning and self-development. We consider ourselves very lucky and honored to have so many wonderful people lead us through in completion of this project.

First and foremost, we would like to thank Dr. Nitin Rakesh, HoD, CSE who gave us an opportunity to undertake this project.

We are grateful to Mr. Shobhit Tyagi for his guidance in our project work. Mr. Shobhit Tyagi, who in spite of being extraordinarily busy with academics, took time out to hear, guide and keep us on the correct path. We do not know where we would have been without his help.

The CSE department monitored our progress and arranged all facilities to make life easier. We choose this moment to acknowledge their contribution gratefully.

Name and signature of Students:

ASHIMA JAIN (2019003789)

ARUP SARKAR (2019000503)

ABSTRACT

Image processing is the procedure of improving an image and extracting some useful information from it. Super-resolution (SR) is the process of upscaling a low-resolution (LR) image to a high-resolution (HR) image. Zooming in on a specific area of interest is required for improved performance in medicine, forensics, pattern recognition, satellite imaging, surveillance, etc. Zooming an image may cause it to become blurry or pixelated. Thus, it is necessary for an image to be of high resolution. In this work we have attempted to perform SR using a deep learning model, namely a Generative Adversarial Network (GAN). A GAN works on the concept of adversarial training and consists of two neural networks, a Generator, and a Discriminator network. These two networks compete against each other. The generator creates an HR image from the LR image, while the discriminator classifies the created image as fake or real. We present a GAN with Inception modules and residual blocks to improve feature extraction, feature learning, image generation and classification processes. For model training, Image Super Resolution dataset from Unsplash has been used. The images have been resized to 96×96 and 384×384 pixels to form the LR and HR images respectively, to perform 4 times ($4\times$) upscaling. The presented network yields satisfactory results.

Keywords- Image Processing, Super Resolution, Deep Learning, Generative Adversarial Network

CONTENTS

TITLE PAGE.....	i
CERTIFICATE.....	ii
ACKNOWLEDGEMENT.....	iii
ABSTRACT.....	iv
LIST OF FIGURES.....	vii
LIST OF ABBREVIATIONS.....	ix
CHAPTER 1: INTRODUCTION.....	11
1.1 PROBLEM STATEMENT.....	11
1.2 PROJECT OVERVIEW.....	11
1.3 EXPECTED OUTCOME.....	13
1.4 HARDWARE & SOFTWARE SPECIFICATIONS.....	13
1.4.1 Hardware Requirements.....	13
1.4.2 Software Requirements.....	13
1.4.3 Libraries Required.....	13
1.4.4 Dataset.....	14
1.5 OTHER NON-FUNCTIONAL REQUIREMENTS.....	14
1.6 REPORT OUTLINE.....	14
CHAPTER 2: LITERATURE SURVEY.....	15
2.1 EXISTING WORK.....	15
2.2 PROPOSED SYSTEM.....	22
2.3 FEASIBILITY STUDY.....	22
2.3.1 Technical Feasibility.....	22
2.3.2 Resource Feasibility.....	22
CHAPTER 3: SYSTEM DESIGN & ANALYSIS.....	23
3.1 PROJECT PERSPECTIVE.....	23
3.2 PERFORMANCE REQUIREMENTS.....	23
3.3 SYSTEM FEATURES.....	23
3.4 METHODOLOGY.....	24
3.4.1 Inception Module.....	24
3.4.2 Residual Block.....	24
3.4.3 Generator Network.....	25
3.4.4 Discriminator Network.....	26
3.4.5 Transfer Learning.....	26
3.4.6 Dataset.....	27
3.4.7 Training.....	27
CHAPTER 4: RESULTS AND OUTPUTS.....	29
4.1 PROPOSED MODEL OUTPUTS.....	29

CHAPTER 5: CONCLUSION & FUTURE SCOPE.....	32
5.1 CONCLUSION.....	32
5.2 FUTURE SCOPE.....	32
REFERENCES.....	33
ANNEXURE 1.....	35
ANNEXURE 2.....	36
ANNEXURE 3.....	37
ANNEXURE 4.....	51

LIST OF FIGURES

Fig. 1.2.1	Block Diagram of Super-Resolution using Generative Adversarial Network	13
Fig. 2.1.1	SR process of SRCNN	15
Fig. 2.1.2	ESPCN with sub-pixel convolutional layer	16
Fig. 2.1.3	Comparison of model architectures of SRCNN and FSRCNN	16
Fig. 2.1.4	Network architecture of VDSR	17
Fig. 2.1.5	Cascading Residual Network	17
Fig. 2.1.6	Generator and Discriminator network of SRGAN	18
Fig. 2.1.7	Architecture of ESRGAN (Developed by modifying SRGAN)	18
Fig. 2.1.8	Model architecture of EnhanceNet	19
Fig. 2.1.9	Generator network architecture of SRFeat	19
Fig. 2.1.10	Discriminator network architecture of SRFeat	20
Fig. 2.1.11	Residual dense block of ESRGAN+	20
Fig. 2.1.12	SR process of RankSRGAN with Siamese-like architecture of Ranker	21
Fig. 2.1.13	Generator and Discriminator network architecture of SRPGAN	21
Fig. 2.1.14	Comparison of traditional convolutional block of a GAN and that of SRPGAN	21
Fig. 3.4.1	Inception module with dimensionality reduction	24
Fig. 3.4.2	Residual block of the proposed model	25
Fig. 3.4.3	Proposed architecture of Generator network	25
Fig. 3.4.4	Proposed architecture of Discriminator network	26
Fig. 3.4.5	Architecture of VGG16	27
Fig. 3.4.6	Sample images from Unsplash dataset	27

Fig. 3.4.7	Training process of discriminator network	28
Fig. 3.4.8	Training process of generator network	28
Fig. 4.1.1	Original HR image	29
Fig. 4.1.2	Original LR image	29
Fig. 4.1.3	Image super-resolved by our model	29
Fig. 4.1.4	Image super-resolved by SRGAN	29
Fig. 4.1.5	Super-resolution results of our model	30
Fig. 4.1.6	Example of SR in medicine	31
Fig. 4.1.7	Example of SR in surveillance	31

LIST OF ABBREVIATIONS

APSIT	Advances in Power, Signal, and Information Technology
CARN	Cascading Residual Network
CNN	Convolutional Neural Network
CV	Computer Vision
DL	Deep Learning
ESPCN	Efficient Subpixel Convolutional Neural Network
ESRGAN	Enhanced Super Resolution Generative Adversarial Network
FSRCNN	Fast Super-Resolution Convolutional Neural Network
GAN	Generative Adversarial Network
HD	High Dimension
HR	High Resolution
LeakyReLU	Leaky Rectified Linear Unit
LR	Low Resolution
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
ReLU	Rectified Linear Unit
ResNet	Residual Network
SR	Super Resolution
SRCNN	Super-Resolution Convolutional Neural Network

SRGAN	Super Resolution Generative Adversarial Network
SRPGAN	Super Resolution Perceptual Generative Adversarial Network
VDSR	Very Deep Super-Resolution

CHAPTER 1: INTRODUCTION

1.1 PROBLEM STATEMENT

Smaller spatial resolution (i.e., size) or degradation due to noise, camera de-focus, blur, image compression, etc., might cause a picture to have a reduced resolution and poor image quality. A high pixel density indicates that an image has a high resolution and is sharper and finer, and so it contains more data and finer details about the original scene.

The number of pixels in an image determine the resolution of the image. An LR image contains less number of pixels. An HR image has a high pixel density and is clearer, sharper, and finer. HR images contain more information (pixels) than was actually contained in the LR images, therefore this subject has grown to be a highly well-known research area. The data processing inequality of information theory states that in whatever way data is processed, information that is not already there cannot be added. This is only possible if an additional source of information is present. SR is possible since a neural network can be trained to generate details based on some previous information that it learns from a dataset of images, i.e., the training dataset. In this way, the features and information added to an image would be consistent with the data processing inequality. SR is a complex and a computationally challenging task. Due to all these reasons image processing researchers find SR to be one of the most fascinating and intriguing research areas.

For better performance in pattern recognition and visual analysis, good quality of photos is required for computer vision (CV) tasks. In medical imaging and diagnosis, good images are significant. Applications like forensic, surveillance, and satellite imaging all call for zooming in on a specific area of interest in the image, necessitating excellent quality.

SR algorithms like nearest neighbor upscaling, bilinear and bicubic interpolation produce a blurry image. Also, despite the usage of faster, quicker, and deeper Convolutional Neural Networks (CNNs) for SR task, one significant issue, i.e., how to obtain the finer texture data when SR is conducted at large upscaling factors remains completely unresolved. Performing SR using deep learning techniques such as a Generative Adversarial Network (GAN) [1] on the low-sized image can restore the finer details of the image and yield a larger size image with high resolution.

1.2 PROJECT OVERVIEW

The procedure of generating a high-resolution (HR) image from a low-resolution (LR) image is known as super resolution (SR). SR is the process of zooming into photos and videos beyond their resolution and estimating an HR image from its equivalent LR image. A higher quality picture can be generated from a lesser quality picture using the SR approach. The fundamental principle of SR is the merging of a series of noisy, blurry, LR pictures to create a higher-resolution image or sequence. Deep learning (DL) is a subset of machine learning techniques that seek to learn data hierarchy representations. It is based on artificial neural networks with representation learning. Recently, SR has received state-of-the-art performance due to the application of potent deep learning

algorithms. Due to its ability of learning and resilience to noise, the use of deep learning approaches in SR has generated a lot of research attention recently. SR is mostly used in microscopy, medical imaging, and image recovery.

Goodfellow et al. [1] introduced GANs in 2014. GANs work based on the concept of adversarial training, i.e., it is made up of two neural networks, namely a generator network and a discriminator network, that oppose each other. The GAN model introduced in 2014 used multilayer perceptron (MLP) network to build the generator and the discriminator. The first GAN to use convolutional layers in the generator and discriminator networks was developed in 2015 [2]. The generator takes in random data and converts it into fake images. The discriminator takes in the fake images as well as the original images from the dataset and tries to differentiate between the fake images and the original images. The generator tries to trick the discriminator by trying to generate better and more realistic images after each epoch [3]. In terms of SR, the generator generates a fake HR image from the LR image. The generator and the discriminator, both tussle against each other. The discriminator tries to “discriminate” between the generated, fake HR image and the original HR image, i.e., it tries to classify and distinguish between them, whereas the generator attempts to “fool” the discriminator by producing better realistic-fake HR images after each epoch, so that the discriminator is unable to decide between the generated and the real images, thus resulting in the generation of improved and enhanced HR images after each epoch.

Szegedy et al. [4] developed Inception module in 2015. The Inception module is an advanced convolutional neural network architecture. They named the incarnation of the inception module proposed in their work as GoogleNet. The inception module is made up of several convolutional layers with varied kernel sizes and max-pooling layer. The outputs of these layers are concatenated into a single input for the next stage. The inception module helps to overcome the difficulty of deciding what filter size to use in the convolutional layers and when to add pooling layers. The inception module uses different kernel sizes and concatenates the depth of all the outputs.

The Residual block [5] is another advanced CNN architecture. It was introduced in 2015. The residual block consists of two paths, a shortcut path, and the main path. The shortcut path is called the skip connection. When very deep neural networks are built resulting in increase in the depth of the network, the problem of vanishing or exploding gradient arises. Skip connections are an alternate shortcut path that help to overcome these problems by bypassing a portion of the neural network's layers and sending the output of one layer as the input to later levels. Skip connections also ensure feature reusability which is important for SR task. By taking shortcuts or skip connections, residual networks are able to tackle the deterioration issue by short-circuiting shallow levels to deep layers.

In this work we have incorporated inception modules and residual blocks into the GAN to build an efficient and enhanced network for image super resolution.

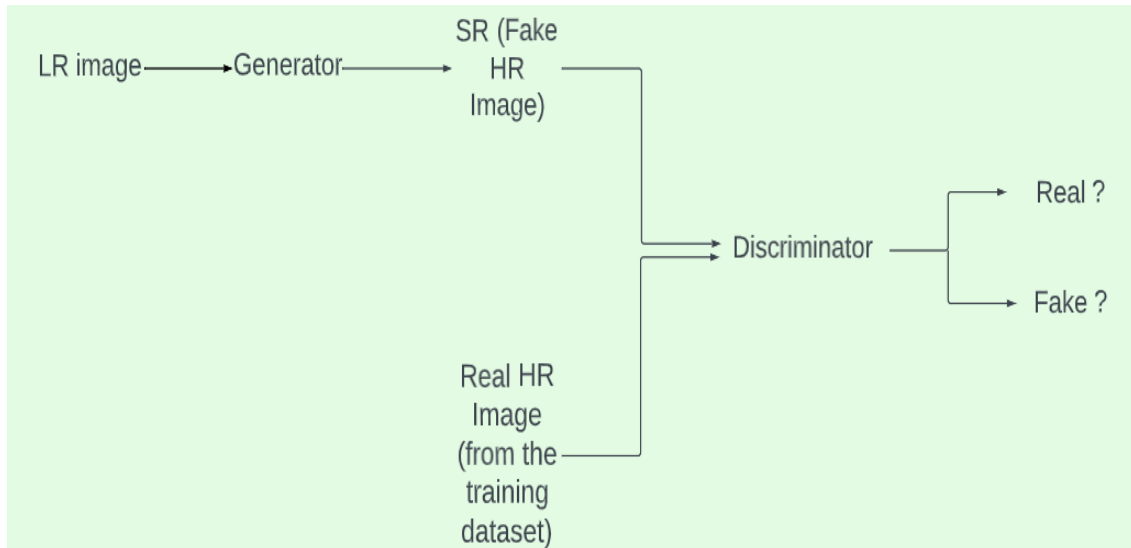


Fig. 1.2.1. Block Diagram of Super-Resolution using Generative Adversarial Network

1.3 EXPECTED OUTCOME

- Research paper submitted to 2nd International Conference on Advances in Power, Signal, and Information Technology (APSIT 2023)
- Acceptance Notification: By 10th May

1.4 HARDWARE & SOFTWARE SPECIFICATIONS

1.4.1 Hardware Requirements

- **CPU:** Intel® Core i5 @1.60 Ghz
- **RAM:** 8GB
- **Storage:** At least 10 GB free storage.
- **Basic I/O:** using generic keyboard and mouse.
- **Display:** Standard monitor

1.4.2 Software Requirements

- **OS:** Windows Operating System
- **Language:** Python
- **IDE:** Jupyter Notebook/Spyder

1.4.3 Libraries Required

- Keras [26]
- Matplotlib [27]
- Tensorflow [28]
- OpenCV [29]
- Tqdm [30]

1.4.4 Dataset

- Unsplash [23]

1.5 OTHER NON-FUNCTIONAL REQUIREMENTS

- **Reliability** - The probability of the software functioning faultlessly for a fixed period is called reliability.
- **Maintainability** - This means how long it usually takes, how easy it is, and how rapidly a software can be fixed or restored after an error occurs.
- **Recoverability** - It is the capability of a software to recuperate from a crash or failure and recommence regular functioning.
- **Performance** - Performance means how quickly a system can respond to a particular user's action while managing a specific workload.
- **Serviceability** - This feature shows how simple it is to provide service when it is required.

1.6 REPORT OUTLINE

- Chapter 2 focuses on the previous work done to highlight the existing models for SR, proposed system, and feasibility of our project.
- Chapter 3 focuses on the proposed model, network architecture, how it proceeds, what are the requirements and system features, and what all will be the methodology to build our model.
- Chapter 4 provides the results of our model.
- Chapter 5 finally concludes the project report and discusses about the future scope of the proposed system.

CHAPTER 2: LITERATURE SURVEY

2.1 EXISTING WORK

SR is the process of combining LR (noisy) sequences of images of a scene to create a HR image or image sequence. As a result, it attempts to rebuild the original HR scene image from a set of LR observed images. It is a subset of image processing algorithms used in CV, with several real-world applications such as medical imaging, surveillance and security. High-quality images contain more information than was actually contained in LR photos, therefore the subject has grown to be a highly well-known research area. SR is a difficult computational problem. SR is one of the most fascinating research areas for image processing experts as a result of all these features. Due to the rapid rise of DL techniques, DL-based SR models have been extensively investigated in recent years, and they regularly achieve cutting-edge performance on many SR benchmarks. DL methods have been utilized to handle SR challenges ranging from primary CNN-based methods to recent promising SR approaches using GAN.

Dong et al. [6] developed the Super-Resolution Convolutional Neural Network (SRCNN) in 2014, which was the first super-resolution model that employed CNN. SRCNN has a straightforward architecture, with only three layers and no connecting or pooling layers [7]. The network is made up of three components: non-linear relationship mapping, picture block extraction and representation, and HR image reconstruction. SRCNN has three convolution layers that implement the functions of the aforementioned three components [8]. Patch extraction, also known as feature extraction, is the first convolutional layer that uses the input pictures to produce feature maps. The non-linear mapping layer, which is the next convolutional layer, transforms the feature-maps into high-dimension (HD) feature vectors. The third convolution layer merges the feature maps to produce the HR image in its entirety [9]. The SRCNN process includes the ensuing steps: 1. Preprocessing: Upscaling the LR image to the appropriate high-quality image size is considered preprocessing. 2. Extraction of features: The upscaled LR image is used to extract a number of feature maps. 3. Nonlinear mapping: The mapping of high-quality patches to feature maps that represent LR. 4. Reconstruction: Creation of excellent images from excellent patches [10].

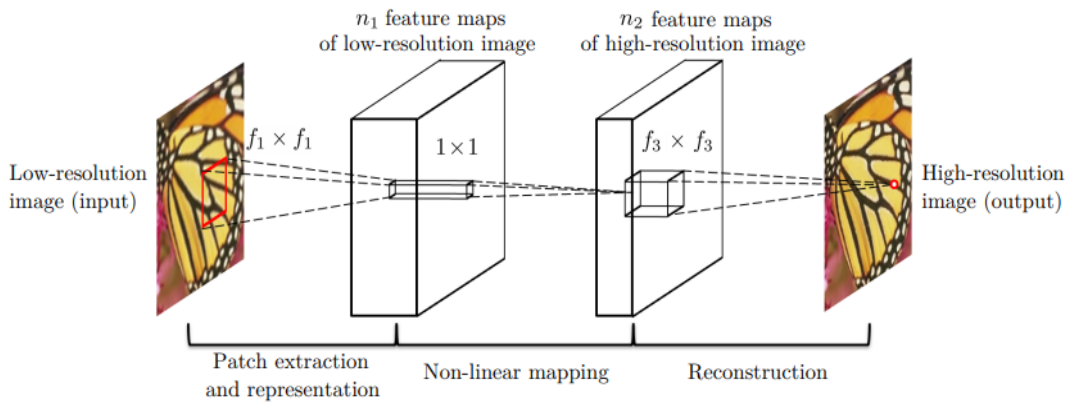


Fig. 2.1.1. SR process of SRCNN [6]

Preprocessing can elevate network parameters, which raises the network's overall computing cost. Shi et al. [11] developed the Efficient Subpixel Convolutional Neural Network (ESPCN) in 2016 that uses a subpixel convolutional layer at the completion to combine LR feature maps and execute a simultaneous estimation of HD space to recreate the HR image. It uses the original LR photos as network inputs without the need for any image preprocessing [8].

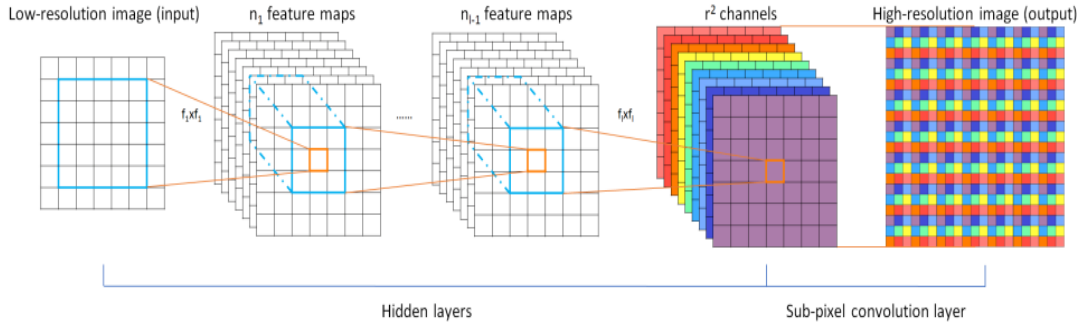


Fig. 2.1.2. ESPCN with sub-pixel convolutional layer [11]

Dong et al. [12] introduced the Fast SRCNN (FSRCNN) in 2016 that accelerates the SRCNN process and enhances the convolutional layer by bypassing bicubic interpolation and switching non-linear mapping for decreasing and expanding layers. A deconvolutional layer is added in FSRCNN towards the completion of the network to allow straight mapping from the LR picture to the HR image with no need of interpolation. The five components of FSRCNN are: extraction of features, LR feature dimension shrinking, LR-HR mapping (non-linear mapping), LR-HR feature expansion, and deconvolution. There are four convolution layers and one deconvolution layer in total. In addition, FSRCNN inserts a convolution layer with a 1×1 convolution kernel to compress LR features before mapping, and then expand them back to their original size after the process is complete. This eliminates the requirement for interpolation preprocessing of LR pictures [8].

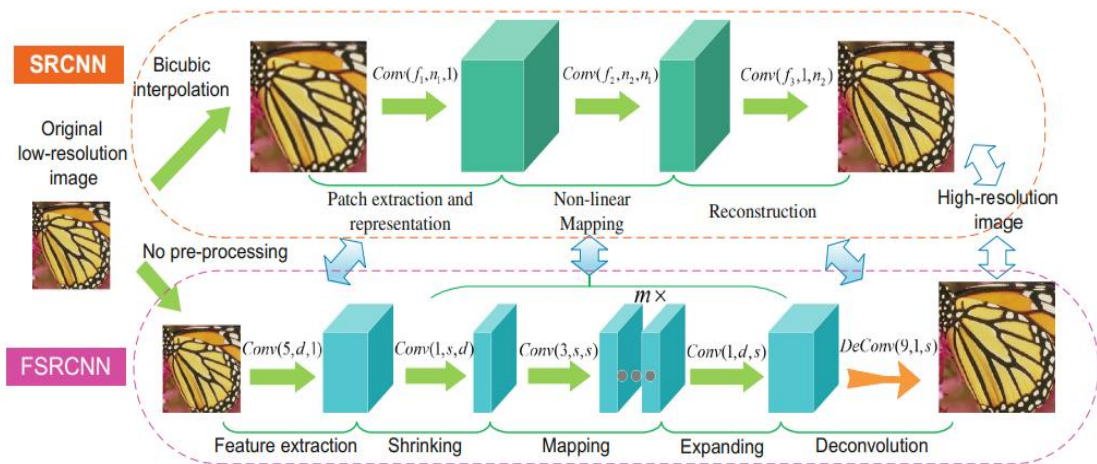


Fig. 2.1.3. Comparison of model architectures of SRCNN and FSRCNN [12]

Kim et al. [13] developed the Very Deep Super-Resolution (VDSR). All network layers of this architecture, also referred to as the VGGnet [24], employ fixed-size (3×3) convolutions. They suggest two efficient methods to lessen learning difficulties and accelerate and improve network convergence. First, the layers memorize a residual mapping that produces the dissimilarity amongst the HR and LR image rather than immediately producing an HR image. Thus, it offers a straightforward goal, and the network solely concentrates on high-frequency data. In addition, gradients are trimmed inside a range, allowing for extremely rapid learning rates that hasten the training process. Their findings lend credence to the idea that deeper networks can learn generalizable representations for multi-scale super-resolution and offer improved contextualization [9].

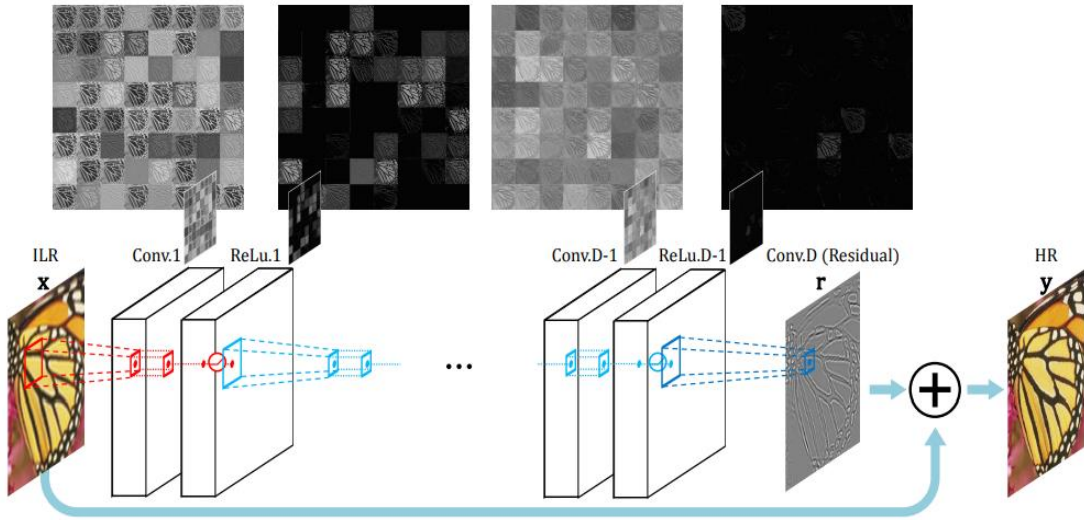


Fig. 2.1.4. Network architecture of VDSR [13]

Ahn et al. [14] presented the Cascading residual network (CARN) that uses Residual Neural Network (ResNet) Blocks to discover the connection among LR and HR input image and output image. The existence of global and local cascading modules distinguishes the two types. An 11 convolutional layer is formed by cascading and converging the features from in-between layers. With the exception of the blocks, which are straightforward residual blocks, global cascading connections are similar to local cascading connections. The multi-level representation and numerous short-cut connections used in this method make information dissemination efficient [9].

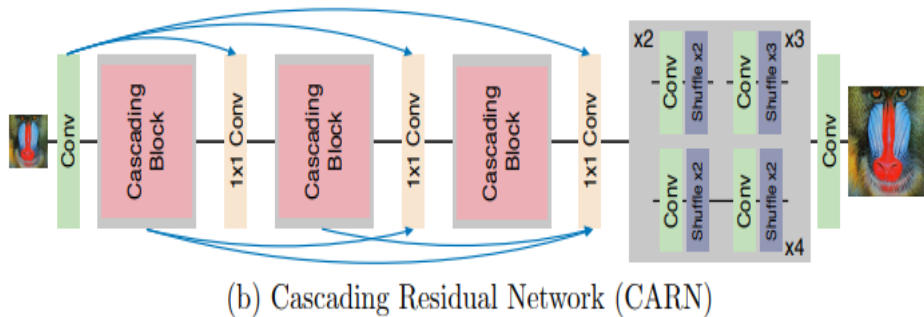


Fig. 2.1.5. Cascading Residual Network [14]

The first SR model to employ GAN is Super-Resolution Generative Adversarial Network (SRGAN) [15]. Using an adversarial objective function that promotes output that is well

resolved and closely resembles the variety of common images is advised by SRGAN. Their formulation of a three-part multitask loss function, which contains an adversarial loss that stabilises a minimum-maximum game between the discriminator and the generator, a perceptual similarity measure precisely defined over HR image depiction, and a Mean Squared Error (MSE) loss that encrypts the similarity per pixel, is their major contribution. In essence, the suggested paradigm favours perceptually similar output to high-dimensional visuals.

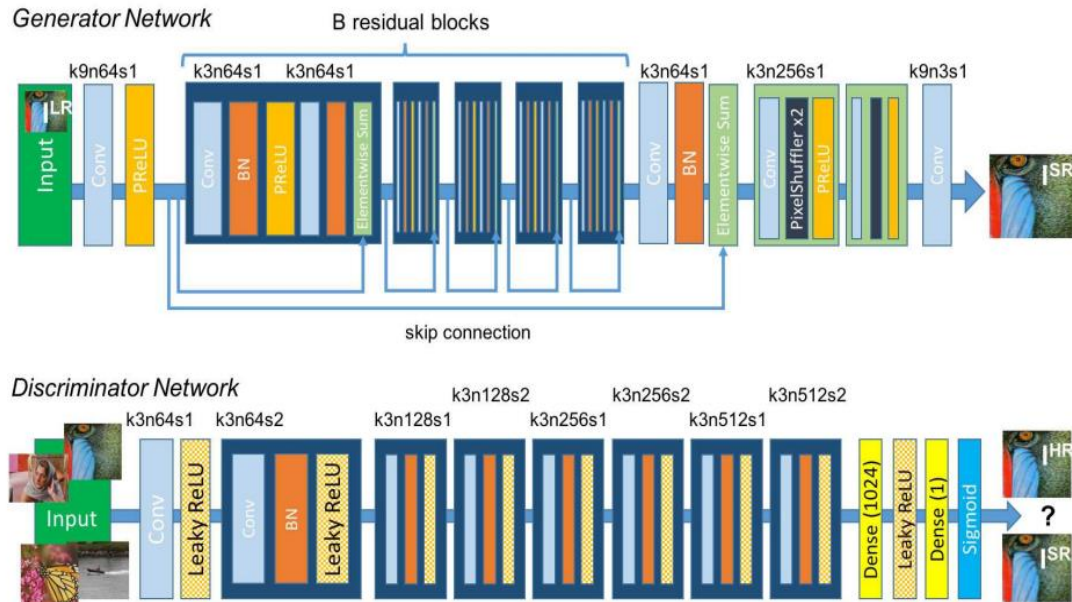


Fig. 2.1.6. Generator and Discriminator network of SRGAN [15]

Wang et al. [16] developed the Enhanced Super-Resolution Generative Adversarial Networks (ESRGAN) in 2018. By removing batch normalization and adding dense blocks, ESRGAN improves on SRGAN. The output from each dense block is connected to the input of the respective block, creating a residual connection over each dense block. In order to enforce residual learning, ESRGAN additionally contains a global residual connection.

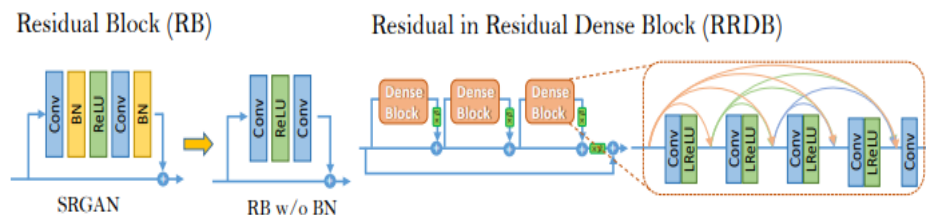


Fig. 2.1.7. Architecture of ESRGAN [16] (Developed by modifying SRGAN)

EnhanceNet [17] employed a GAN and was developed in 2017. Its primary goal was to provide accurate texture features in super-resolved, high-resolution photographs. Apart from the conventional pixel-level MSE loss, EnhanceNet also employed the following two terms for loss: (a) For the intermediary feature presentation of a pre-trained network,

the perceptual loss function was defined as l1 distance. (b) The texture matching loss is measured as the l1 loss among gram matrices generated from deep features and is used to match the texture of LR and HR images.

Output size	Layer
$w \times h \times c$	Input I_{LR}
$w \times h \times 64$	Conv, ReLU
	Residual: Conv, ReLU, Conv
	...
$2w \times 2h \times 64$	2x nearest neighbor upsampling Conv, ReLU
$4w \times 4h \times 64$	2x nearest neighbor upsampling Conv, ReLU
	Conv, ReLU
	Conv
$4w \times 4h \times c$	Residual image I_{res}
	Output $I_{est} = I_{bicubic} + I_{res}$

Fig. 2.1.8. Model architecture of EnhanceNet [17]

SRFeat [18] is another GAN based SR model that is based on feature discrimination. SRFeat focusses on generating high frequency structural characteristics instead of noisy artefacts from the input image utilising a supplementary discriminator that helps the generator do so. By separating the characteristics of generated and actual images, this requirement is met. To extract features, this network has a 9×9 convolutional layer. Then, 1×1 convolutional residual blocks with long-range skip connections are employed. To obtain the desired output size, pixel shuffler layers up-sample the feature maps. The network consists of 16 residual blocks with 64 and 128 feature map settings. An amalgamation of adversarial or perceptual loss and pixel-level loss functions is used that is improved with an Adam optimizer [9].

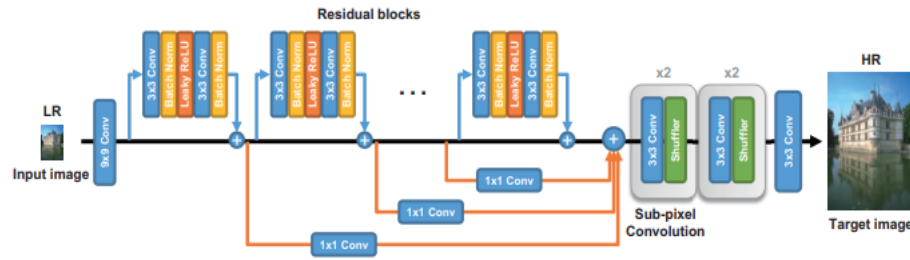


Fig. 2.1.9. Generator network architecture of SRFeat [18]

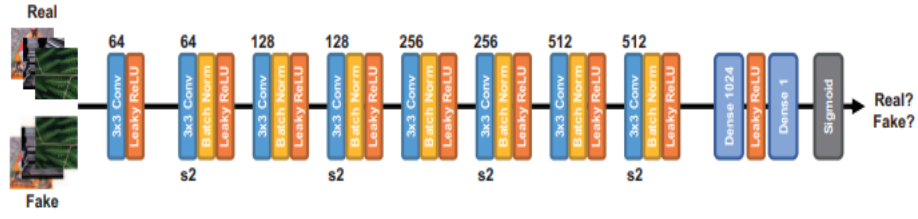


Fig. 2.1.10. Discriminator network architecture of SRFeat [18]

ESRGAN+ [19] presented a new basic block replacing the one used in ESRGAN [16]. Noise inputs were also introduced to the generator network to utilize stochastic variation. An extra level of residual learning was added to the original residual-in-residual dense block presented in ESRGAN. A residual was also added after every 2 layers in each dense block.

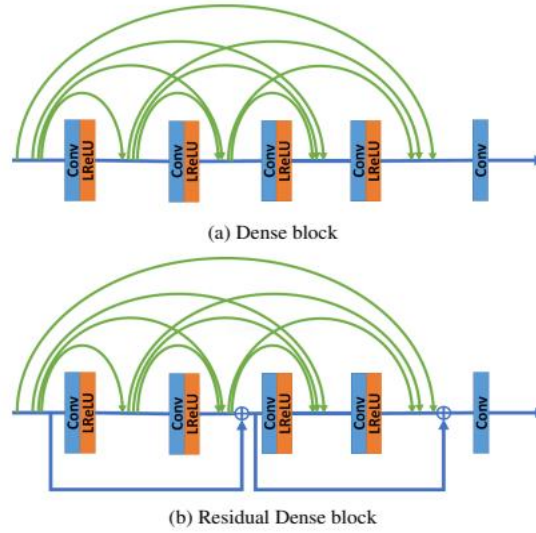


Fig. 2.1.11. Dense block used in ESRGAN(Top). Residual Dense Block used in ESRGAN+(Bottom) [19].

RankSRGAN [20] employed a Ranker model in the SRGAN [15]. The ranker is able to comprehend the perceptual metrics and is based on the Siamese architecture. They also introduced a new rank-content loss to enhance the perceptual quality of the images. The trained ranker is able to rank images as per their perceptual scores. A succession of convolutional, LeakyReLU, pooling, and full-connected layers make up the ranker's two identical network branches. After the Feature Extractor, a Global Average Pooling layer is used. A fully-connected layer is employed as a regressor to quantify the rank outcomes and provide the ranking scores.

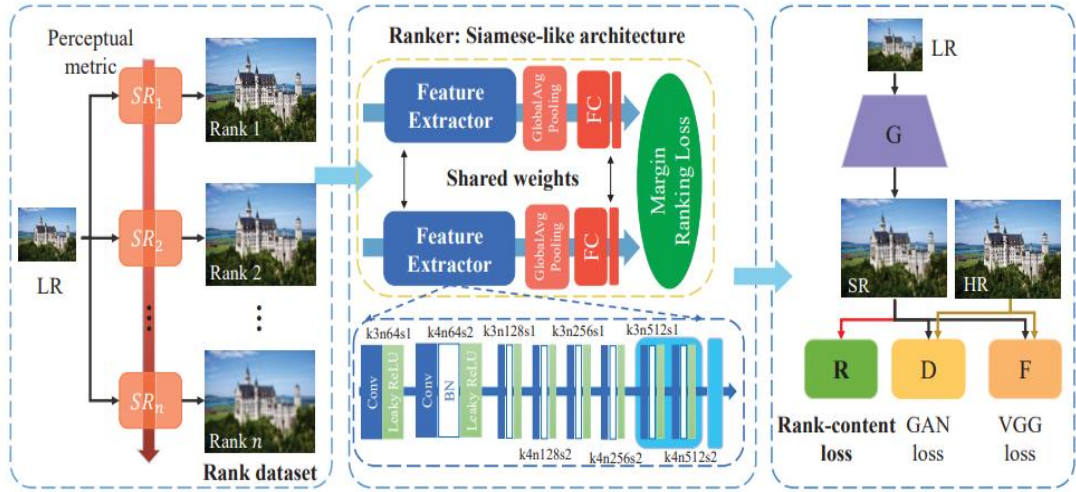


Fig. 2.1.12. SR process of RankSRGAN with Siamese-like architecture of Ranker [20]

The super resolution perceptual generative adversarial network (SRPGAN) [21] is based on the Image-to-Image model [22]. They used the features extracted by the discriminator network to build the perceptual loss and the Charbonnier loss function as the content loss. The batch normalization layer was replaced by the instance normalization layer.

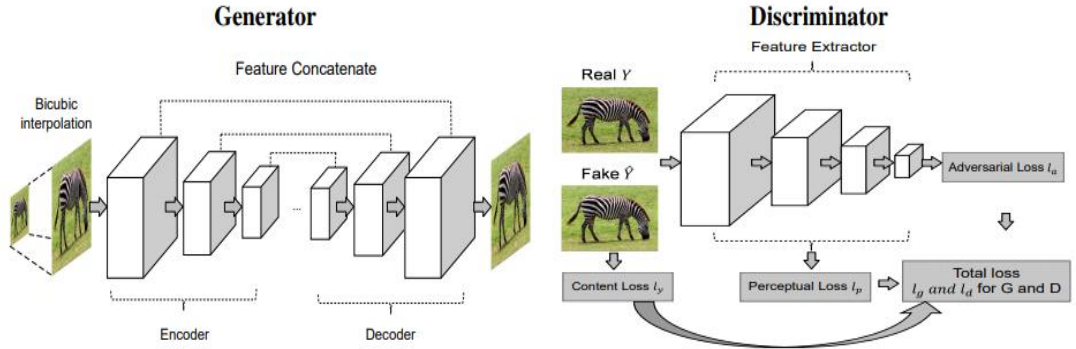


Fig. 2.1.13. Generator and Discriminator network architecture of SRPGAN [21]

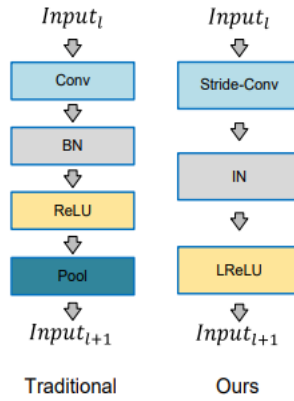


Fig. 2.1.14. Comparison of traditional convolutional block of a GAN and that of SRPGAN [21]

2.2 PROPOSED SYSTEM

In spite of the use of faster, quicker and deeper CNNs for performing SR task, one major issue is still completely unresolved, i.e., how to retrieve the finer texture features when SR is performed at high upscaling factors. GANs are highly efficient for performing SR as they can preserve the finer details of the image and generate a sharper and enhanced HR image for high upscaling factors.

The proposed model is a generative adversarial network with inception modules and residual blocks in the generator and discriminator networks. It is an efficient and enhanced network for image super resolution.

The Unsplash dataset [23] has been used for training the network. The model has been trained on 1,000 LR and HR images from Unsplash. The images have been resized to 96×96 and 384×384 pixels to form the LR and HR images respectively, to perform 4 times upscaling.

2.3 FEASIBILITY STUDY

2.3.1 Technical feasibility: The main technologies and tools associated with the project are:

- Any standard PC or laptop
- Windows Operating System or Ubuntu or MacOS.
- Python 3.0 or above, Keras, tensorflow, matplotlib, OpenCV.
- Python environment installed.
- The technical abilities needed are manageable, and all these libraries and tools are freely accessible.

Therefore, the proposed model is technically feasible.

2.3.2 Resource feasibility: Resources that are essential for the project include:

- A programming device like a laptop or a computer.
- Programming tools that are freely available.
- Programming individuals

Therefore, the model has the required resource feasibility.

CHAPTER 3: SYSTEM DESIGN & ANALYSIS

3.1 PROJECT PERSPECTIVE

Image processing is the procedure of improving an image and extracting some useful information from it. It basically takes an image as input and generates image features or an image itself. It is one of the most significantly growing topics in computer science. DL is based on artificial neural networks. In numerous artificial intelligence fields, including computer vision, speech recognition, and natural language processing, deep learning has demonstrated a clear advantage over traditional machine learning techniques. In general, the development of effective computer technology and the advancement of complex algorithms are responsible for the strong ability of deep learning to handle significant unstructured data. Due to the recent application of potent deep learning techniques, SR has shown improved performance.

This project aims to accomplish SR, which creates a HR image from a LR image. The purpose of the famously challenging work of SR is to produce an HR output from its LR variant. Recently, SR has demonstrated better performance as a result of the use of powerful DL algorithms. The application of DL techniques in super-resolution problems has lately attracted a lot of research attention due to its capacity for learning and resilience to noise. Medical imaging, image recovery, and microscopy are the main applications for deep learning super resolution. Despite the usage of faster, quicker, and deeper CNNs for conducting SR tasks, one significant issue, namely, how to obtain the finer texture data when SR is conducted at large upscaling factors, remains completely unanswered. A highly effective network for conducting SR is the GAN, which can preserve the image's finer details and provide a sharper, improved HR image for large upscaling factors. High-quality images contain more information than was actually contained in LR photos, therefore the subject has grown to be a highly well-known research area. Super-resolution is a computationally difficult and numerically unresolved subject. It is one of the most fascinating research areas for those studying image processing as a result of all these aspects.

3.2 PERFORMANCE REQUIREMENTS

- **Speed and Latency Requirements-** Determines the time needed to execute certain tasks. These specifications frequently mention response times. In approximately 3 seconds, the pre-trained model should generate the HR image from the LR image.
- **Precision or Accuracy Requirements-** It specifies the required degree of accuracy in the outcome of the software. The model should produce images with enhanced features that are perceptually decent.

3.3 SYSTEM FEATURES

The Unsplash dataset, that is a high-quality image dataset has been used to train the model. This dataset was recently presented for image restoration applications. The model has been trained on 1,000 LR and HR images from Unsplash. The images have been

resized to 96×96 and 384×384 pixels to form the LR and HR images respectively, to perform 4 times upscaling. The model has been trained for 500 epochs. The trained model can super-resolve an image in approximately 3 seconds.

3.4 METHODOLOGY

3.4.1 Inception Module

The inception module presented in this work consists of 1×1 convolutional layers before the 3×3 , 5×5 convolutional layers and after the 3×3 max-pooling layer as shown in Fig. 16. This is an inception module with dimensionality reduction. These 1×1 convolutional layers are called reduce layers as they help to lessen the computational cost.

The module essentially functions as numerous convolution filters are applied to the same input with some pooling. The output is then concatenated. This enables the model to use multi-level feature extraction. For example, it extracts both general 5×5 and local 1×1 characteristics simultaneously. Using several features from various filters improves network performance.

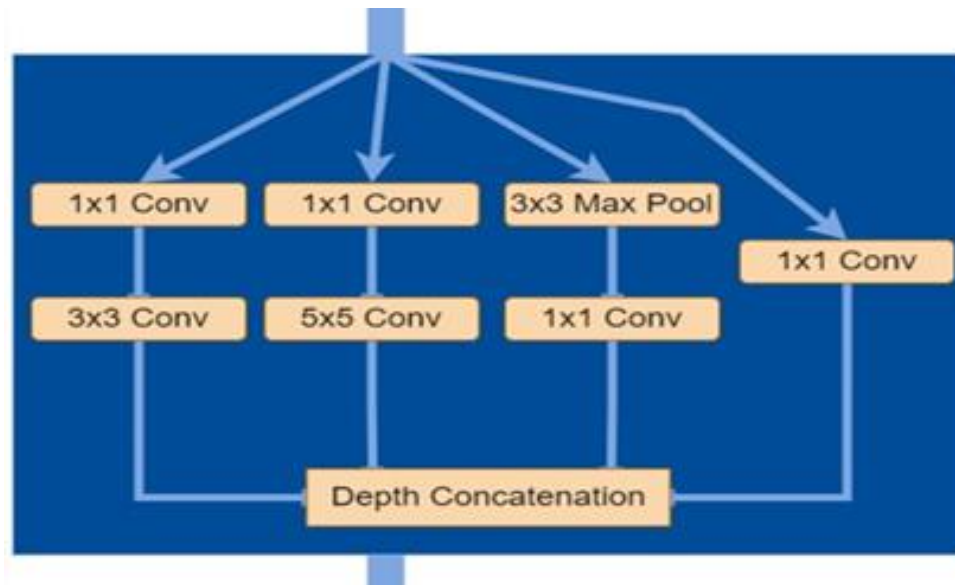


Fig. 3.4.1. Inception module with dimensionality reduction

3.4.2 Residual Block

The residual block consists of a main path and a shortcut path. The main path consists of convolutional layers and batch normalization layers to reduce overfitting and to speed up the training. The first two convolutional layers consists of ReLU activations after batch normalization. The shortcut path that contains the output of the previous layers is added to the main path as shown in Fig. 17.

Skip connections were introduced in 2015 [5]. They are added to overcome the degradation problem of the neural networks. The performance of the model suffers as the depth of the architecture rises while training deep neural networks. This is the deterioration problem. The causes could be overfitting, in which the model tends to overfit

as depth increases, or vanishing gradient and/or inflating gradient issues. Skip connections, which bypass a portion of the neural network's layers and send the output of one layer as the input to succeeding levels, are used to address this problem. Skip connections also ensure feature reusability which is important for super resolution task. Concatenation aims to incorporate traits discovered in earlier layers into deeper layers as well. Performance deterioration is common in very deep networks. By taking shortcuts or skip connections, residual networks are able to tackle the deterioration issue by short-circuiting shallow levels to deep layers. Residual blocks can then be stacked infinitely high with no performance degradation. This makes it possible to create very deep networks.

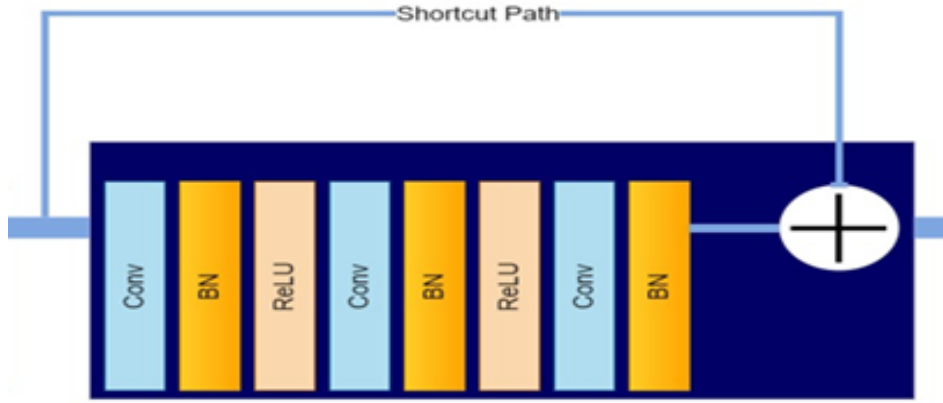


Fig. 3.4.2. Residual block of the proposed model

3.4.3 Generator Network

The generator is an inverted CNN that starts with an LR image and upscales the image until the required image dimension is achieved. It consists of convolutional layers with ReLU activation function, followed by 10 residual blocks with skip connections and ReLU activation function layers in between. It then contains 4 inception modules stacked as follows: 1 inception module + 3×3 max-pooling layer + 2 inception modules + 3×3 max-pooling layer + 1 inception module. It then consists of a convolutional layer followed by batch normalization. This is followed by upscaling blocks to upscale the image 4 times, and a convolutional layer at the end. The generator network architecture is shown in Fig. 18.

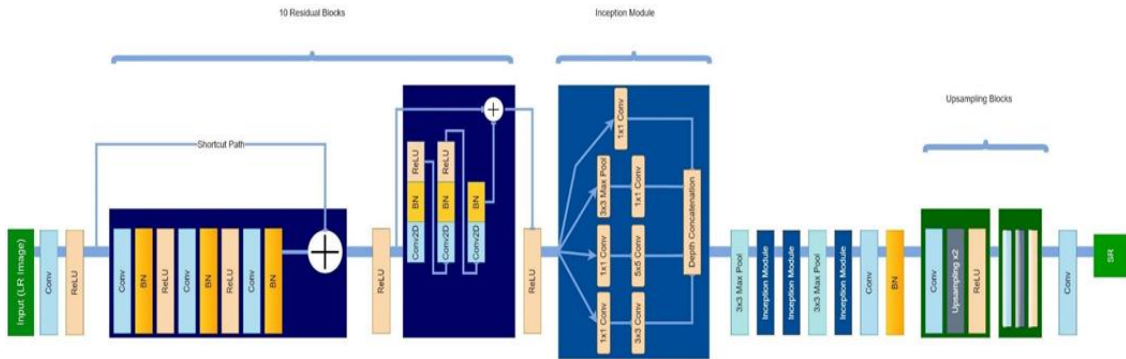


Fig. 3.4.3. Proposed architecture of Generator network

3.4.4 Discriminator Network

The discriminator network is a conventional CNN in which the convolutional layers reduce in size until they reach the flatten layer. It consists of 5 discriminator blocks. The discriminator block consists of convolutional layer with batch normalization and LeakyReLU activation function. It then consists of 4 inception modules stacked as follows: 1 inception module + 3×3 max-pooling layer + 2 inception modules + 3×3 max-pooling layer + 1 inception module, followed by 2 more discriminator blocks. The next part of the discriminator network is the fully connected classifier which classifies the images as fake (generated by the generator) or original HR image. The classifier part consists of a flatten layer to flatten the 2D image matrix into one long vector with dimension $(1, n)$, where n is the total number of pixels in the image, a dense or fully-connected layer followed by LeakyReLU activation function and a dense layer with sigmoid activation function to categorize the image as real or fake. The architecture is shown in Fig. 19.

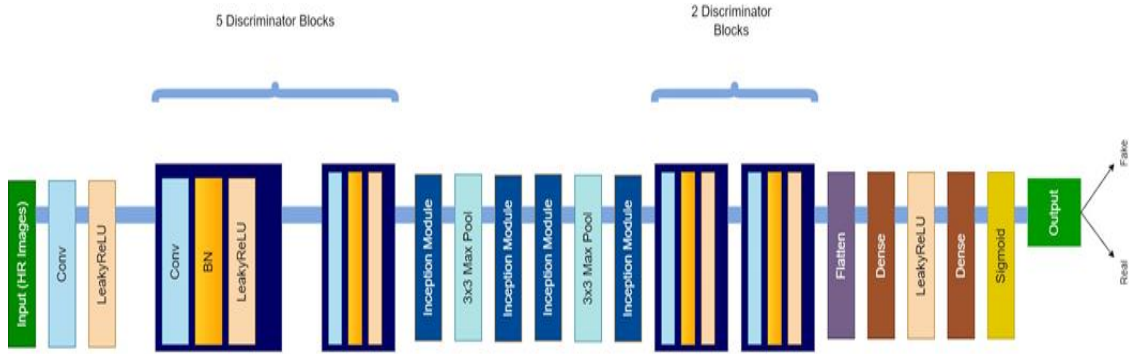


Fig. 3.4.4. Proposed architecture of Discriminator network

3.4.5 Transfer Learning

Transfer learning is the process of applying what a neural network has learned from being trained on a specific dataset to another problem. The weights in reused layers can be used as an opening point for the training procedure and can then be modified as per the new problem. Transfer learning is flexible, enabling pre-trained models to be used directly as a feature extractor or can be combined with entirely new models. A model that has been trained on a sizable benchmark dataset to solve a problem similar to the problem in hand is said to be pre-trained. In this work, VGG16 [24] network has been used as a feature extractor. It was pre-trained on ImageNet [25]. To extract meaningful features from new samples, the feature maps learned by this network were reused, and a new classifier was added on top of it. VGG was developed in 2014 by the Visual Geometry Group at Oxford University. VGG16 network consists of 16 weight layers (13 convolutional layers and 3 dense layers), as shown in Fig. 20.



Fig. 3.4.5. Architecture of VGG16 [3]

3.4.6 Dataset

More than 250k photographers contributed to the Unsplash dataset [23], which also includes information from billions of searches conducted across different contexts and applications.

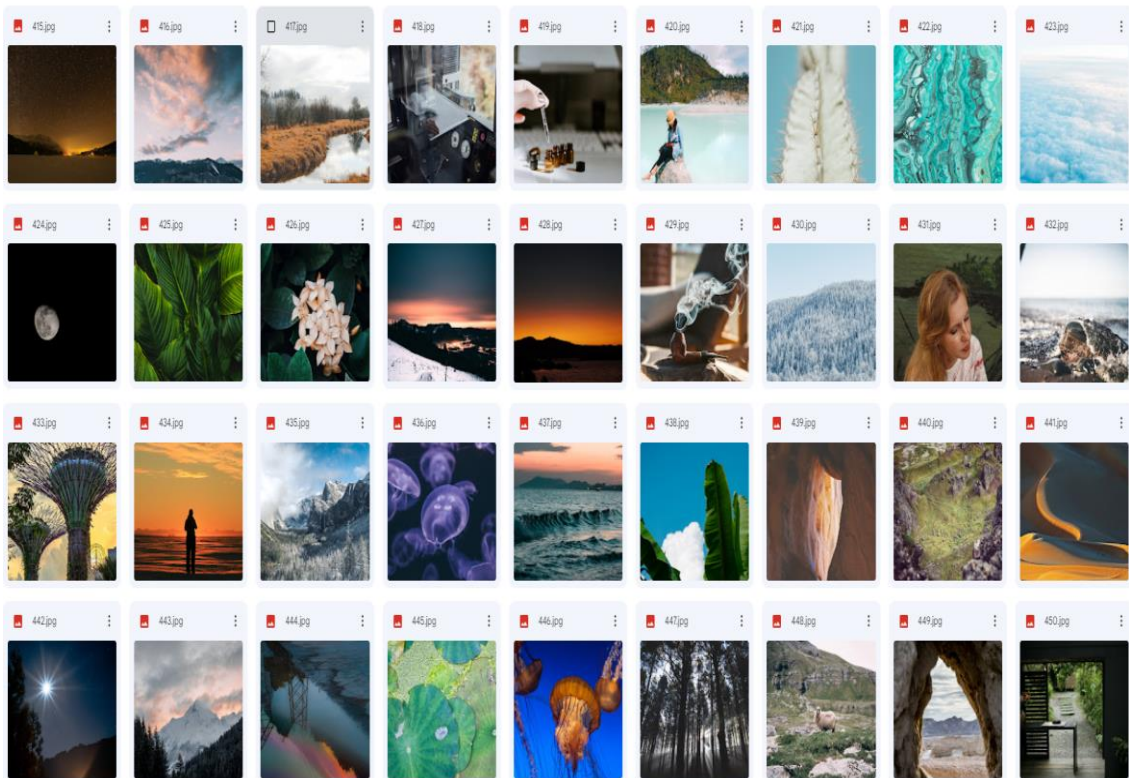


Fig. 3.4.6. Sample images from Unsplash dataset

3.4.7 Training

Training the discriminator network is a straightforward supervised training process. The network learns to discriminate between real and fake images with the help of labelled images from the generator (fake) and training data (original). The training process of discriminator is shown in Fig. 22.

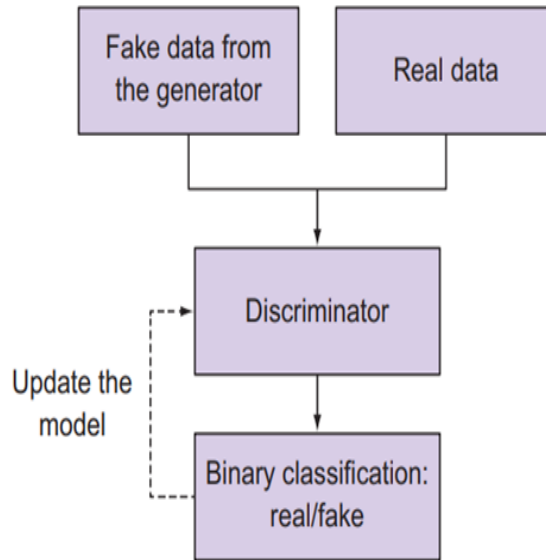


Fig. 3.4.7. Training process of discriminator network [3].

Unlike the discriminator, the generator cannot be trained by itself. It requires the discriminator model to determine whether or not it successfully faked images. Hence, using both the models, the generator and the discriminator, a combined network is created to train the generator. While training the generator the weights of the discriminator are frozen. The training process of generator is shown in Fig. 23.

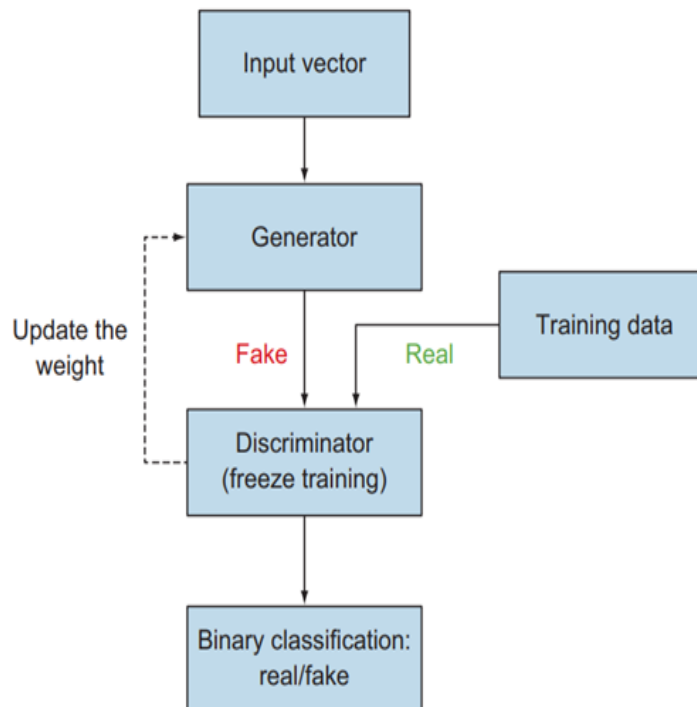


Fig. 3.4.8. Training process of generator network [3].

CHAPTER 4: RESULTS AND OUTPUTS

4.1 PROPOSED MODEL OUTPUTS

The proposed model performs well. The subjective findings are shown in the figures below.



Fig. 4.1.1. Original HR image



Fig. 4.1.2. Original LR image



Fig. 4.1.3. Image super-resolved by our model



Fig. 4.1.4. Image super-resolved by SRGAN [15]

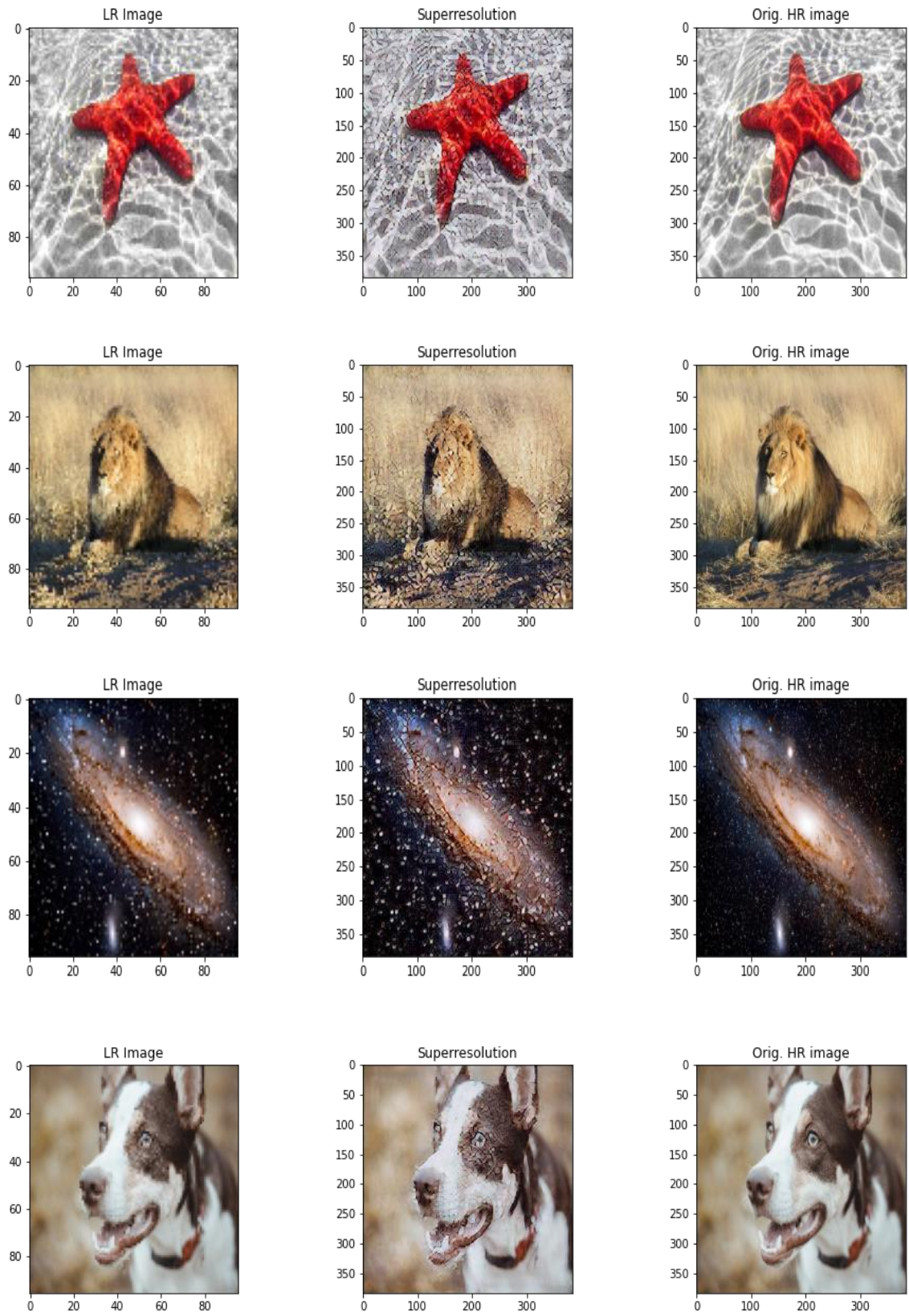


Fig. 4.1.5. Results of the presented model

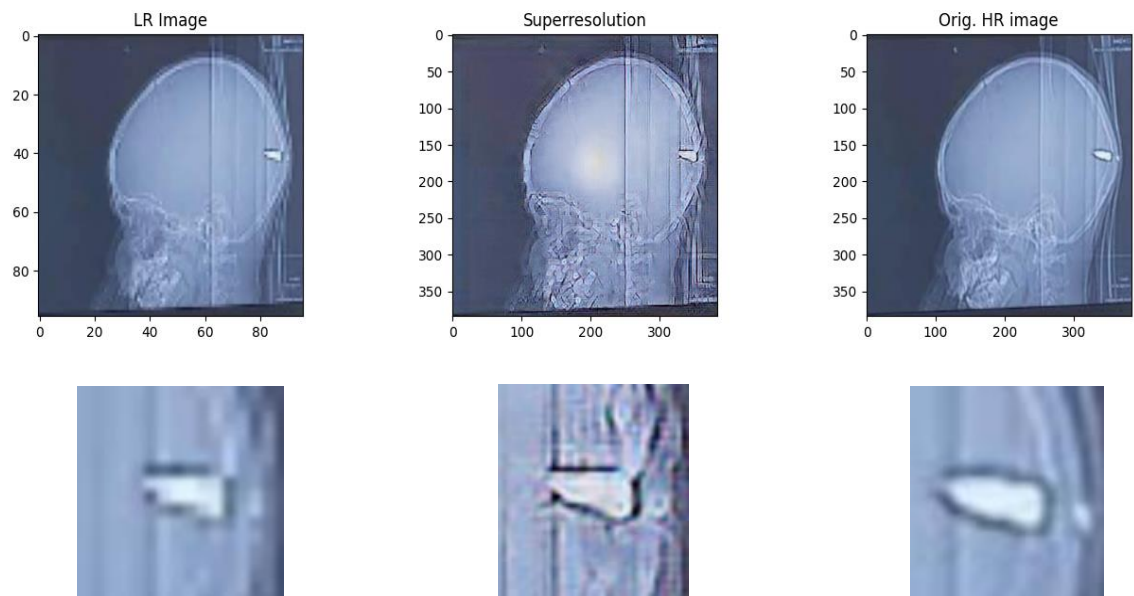


Fig. 4.1.6. Example of SR in medicine

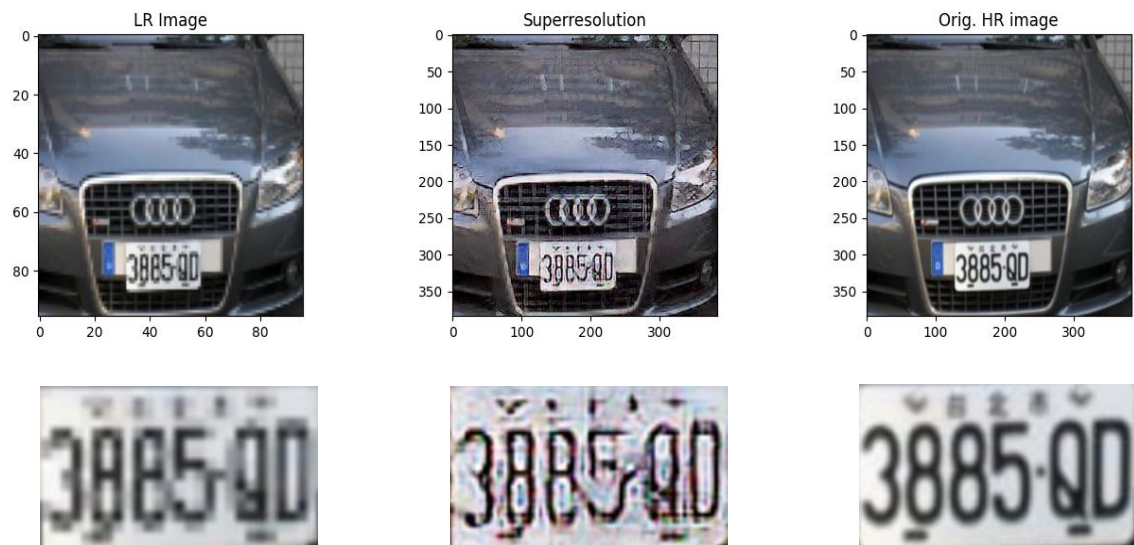


Fig. 4.1.7. Example of SR in surveillance

CHAPTER 5: CONCLUSION & FUTURE SCOPE

5.1 CONCLUSION

A Generative Adversarial Network with Inception modules and Residual blocks in the generator and discriminator networks for image super resolution has been presented in this work. The network has been trained on images from the Unsplash dataset and can perform 4 times upscaling. The presented GAN with inception and residual blocks performs well and gives satisfactory qualitative results.

5.2 FUTURE SCOPE

- We intend to investigate the network to give better results with different upscaling factors by increasing the training data and training the model for more epochs to obtain better results.
- The network architecture can be further improved by adding more skip connections.
- The model can be further deployed by building a proper GUI such as a website so that it can be used more efficiently and easily.
- Currently, resizing the image to 96x96 and 384x384 pixels is required for super-resolution. The model needs to be modified so that no specific resizing is required and image of any size can be super-resolved.



REFERENCES

- [1] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11), 139-144.
- [2] Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.
- [3] Elgendy, M. (2020). *Deep learning for vision systems*. Simon and Schuster.
- [4] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).
- [5] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
- [6] Dong, C., Loy, C. C., He, K., & Tang, X. (2014). Learning a deep convolutional network for image super-resolution. In *Computer Vision—ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV 13* (pp. 184-199). Springer International Publishing.
- [7] Cao, Z., Liu, X., & Wang, Z. (2022, May). Single image super-resolution via deep learning. In *2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA)* (pp. 425-430). IEEE.
- [8] Wang, W., Hu, Y., Luo, Y., & Zhang, T. (2020). Brief survey of single image super-resolution reconstruction based on deep learning approaches. *Sensing and Imaging*, 21, 1-20.
- [9] Anwar, S., Khan, S., & Barnes, N. (2020). A deep journey into super-resolution: A survey. *ACM Computing Surveys (CSUR)*, 53(3), 1-34.
- [10] Kumar, A. (2021). Super-Resolution with Deep Learning Techniques: A Review. *Computational Intelligence Methods for Super-Resolution in Image Processing Applications*, 43-59.
- [11] Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., ... & Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1874-1883).
- [12] Dong, C., Loy, C. C., & Tang, X. (2016). Accelerating the super-resolution convolutional neural network. In *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14* (pp. 391-407). Springer International Publishing.
- [13] Kim, J., Lee, J. K., & Lee, K. M. (2016). Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1646-1654).
- [14] Ahn, N., Kang, B., & Sohn, K. A. (2018). Fast, accurate, and lightweight super-resolution with cascading residual network. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 252-268).
- [15] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., ... & Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4681-4690).
- [16] Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., ... & Change Loy, C. (2018). Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European conference on computer vision (ECCV) workshops*.
- [17] Sajjadi, M. S., Scholkopf, B., & Hirsch, M. (2017). Enhancenet: Single image super-resolution through automated texture synthesis. In *Proceedings of the IEEE international conference on computer vision* (pp. 4491-4500).
- [18] Park, S. J., Son, H., Cho, S., Hong, K. S., & Lee, S. (2018). Srfeat: Single image super-resolution with feature discrimination. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 439-455).

- [19] Rakotonirina, N. C., & Rasoanaivo, A. (2020, May). ESRGAN+: Further improving enhanced super-resolution generative adversarial network. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 3637-3641). IEEE.
- [20] Zhang, W., Liu, Y., Dong, C., & Qiao, Y. (2019). Ranksrgan: Generative adversarial networks with ranker for image super-resolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 3096-3105).
- [21] Wu, B., Duan, H., Liu, Z., & Sun, G. (2017). SRPGAN: perceptual generative adversarial network for single image super resolution. *arXiv preprint arXiv:1712.05927*.
- [22] Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1125-1134).
- [23] Unsplash, "Access the world's largest open library dataset for free.," Unsplash Dataset | The world's largest open library dataset. [Online]. Available: <https://unsplash.com/data>. [Accessed: 20-Feb-2023].
- [24] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- [25] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248-255). Ieee.
- [26] Chollet, F., & others. (2015). Keras. GitHub. Retrieved from <https://github.com/fchollet/keras>.
- [27] Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in science & engineering*, 9(03), 90-95.
- [28] Abadi, M. (2016, September). TensorFlow: learning functions at scale. In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming* (pp. 1-1).
- [29] Bradski, G., & Kaehler, A. (2000). OpenCV. *Dr. Dobb's journal of software tools*, 3(2).
- [30] da Costa-Luis, C. O. (2019). tqdm: A fast, extensible progress meter for python and cli. *Journal of Open Source Software*, 4(37), 1277.

ANNEXURE 1

Research paper submission details (8th sem)

APSIT 2023 submission 4606  

APSIT 2023 <apsit2023@easychair.org>

to me ▼

Dear authors,

We received your submission to **APSIT** 2023 (Advances in Power, Signal, and Information Technology):

Authors : Ashima Jain, Arup Sarkar, Pranav Dass and Arun Prakash Agrawal

Title : Generative Adversarial Network with Inception and Residual blocks for Image Super Resolution

Number : 4606

The submission was uploaded by Ashima Jain

<ashimajain2001@gmail.com>. You can access it via the **APSIT** 2023

EasyChair Web page

<https://easychair.org/conferences/?conf=apsit2023>

Thank you for submitting to **APSIT** 2023.

Best regards,

EasyChair for **APSIT** 2023.

ANNEXURE 2

7th sem outcome details

The screenshot shows the IEEE Xplore website interface. At the top, there's a navigation bar with links to IEEE.org, IEEE Xplore, IEEE SA, IEEE Spectrum, and More Sites. On the right, there are links for SUBSCRIBE, Cart (1), Welcome Ashima Jain, and Sign Out. Below the navigation bar, the IEEE Xplore logo is on the left, and a search bar with a dropdown menu set to 'All' is in the center. To the right of the search bar is an 'Institutional Sign In' button. Below the search bar, there's a link to 'ADVANCED SEARCH'.

The main content area displays the title of the paper: 'Improved Generative Adversarial Network for Generating High-Resolution Images from Low-Resolution Images'. Below the title, it says 'Publisher: IEEE' and provides options to 'Cite This' or download a 'PDF'. The authors listed are Ashima Jain, Arup Sarkar, and Arun Prakash Agrawal, with a link to 'All Authors'. There are icons for 'Full Text Views' (6), 'Full Text', and 'Text Views'. To the right of the authors, there are icons for 'R', 'Share', 'C', 'F', and 'B'.

The abstract section is titled 'Abstract:' and contains the following text: 'The procedure of generating a high-resolution (HR) image from a low-resolution (LR) image is known as super-resolution (SR). In this paper, we try to perform SR using Deep Learning techniques. For better performance in medicinal imaging, forensics, pattern recognition, satellite imaging, surveillance, etc., zooming of a particular area of attention in the image is required, making high resolution necessary. We present ISRGAN (Improved Super Resolution Generative Adversarial Network), an improved version of SRGAN (Super Resolution Generative Adversarial Network) for image SR. For training the network, images from the DIV2K dataset have been used. The dataset consists of 800 different images which have been resized to 32x32 and 128x128 pixels to form LR and HR images respectively. A Generative Adversarial Network is based on the idea of Adversarial training, and comprises of 2 parts, a discriminator network, and a generator network. The generator produces a HR image from the LR image, whereas the discriminator network classifies the generated image as fake or real. The presented model produces decent results and our final super-resolved results show that the presented ISRGAN model produces images with enhanced features.'

Below the abstract, there's a 'Published in:' section stating '2023 13th International Conference on Cloud Computing, Data Science & Engineering (Confluence)'. It also provides the 'Date of Conference: 19-20 January 2023', 'Date Added to IEEE Xplore: 22 February 2023', 'DOI: 10.1109/Confluence56041.2023.10048886', 'Publisher: IEEE', and 'Conference Location: Noida, India'. There is also a link to 'ISBN Information:'.

On the right side of the page, there's a 'More Like This' section with a list of related papers, including 'MSTCGAN: Multiscale Time Conditional Generative Adversarial Network for Long-Term Satellite Image Sequence Prediction' and 'Semi-Supervised Deep Learning Seismic Impedance Inversion Using Generative Adversarial Networks'. There is a 'Show More' link at the bottom of this section.

At the bottom right, there's a blue box with the text 'Need Full-Text access to IEEE Xplore for your organization? CONTACT IEEE TO SUBSCRIBE >'.

Online published paper link- <https://ieeexplore.ieee.org/document/10048886>

ANNEXURE 3

Implementation screenshots

Training Epochs

0%		0/720 [00:00<?, ?it/s]1/1 [=====] - 0s 400ms/step
0%		1/720 [00:05<1:04:11, 5.36s/it]1/1 [=====] - 0s 16ms/step
0%		2/720 [00:05<28:44, 2.40s/it] 1/1 [=====] - 0s 17ms/step
0%		3/720 [00:06<17:27, 1.46s/it]1/1 [=====] - 0s 15ms/step
1%		4/720 [00:06<12:10, 1.02s/it]1/1 [=====] - 0s 21ms/step
1%		5/720 [00:06<10:07, 1.18it/s]1/1 [=====] - 0s 32ms/step
1%		6/720 [00:07<08:28, 1.40it/s]1/1 [=====] - 0s 28ms/step
1%		7/720 [00:07<07:29, 1.59it/s]1/1 [=====] - 0s 33ms/step
1%		8/720 [00:08<07:21, 1.61it/s]1/1 [=====] - 0s 25ms/step
1%		9/720 [00:08<07:04, 1.68it/s]1/1 [=====] - 0s 59ms/step
1%		10/720 [00:09<07:11, 1.64it/s]1/1 [=====] - 0s 49ms/step
2%		11/720 [00:10<06:54, 1.71it/s]1/1 [=====] - 0s 14ms/step
2%		12/720 [00:10<05:58, 1.97it/s]1/1 [=====] - 0s 15ms/step
2%		13/720 [00:10<05:21, 2.20it/s]1/1 [=====] - 0s 19ms/step
2%		14/720 [00:11<04:55, 2.39it/s]1/1 [=====] - 0s 15ms/step
2%		15/720 [00:11<04:39, 2.52it/s]1/1 [=====] - 0s 18ms/step
2%		16/720 [00:11<04:23, 2.67it/s]1/1 [=====] - 0s 15ms/step
2%		17/720 [00:12<04:12, 2.78it/s]1/1 [=====] - 0s 15ms/step
2%		18/720 [00:12<04:06, 2.85it/s]1/1 [=====] - 0s 16ms/step
3%		19/720 [00:12<04:01, 2.90it/s]1/1 [=====] - 0s 14ms/step
3%		20/720 [00:13<03:59, 2.92it/s]1/1 [=====] - 0s 16ms/step
3%		21/720 [00:13<03:57, 2.95it/s]1/1 [=====] - 0s 15ms/step
3%		22/720 [00:13<03:56, 2.95it/s]1/1 [=====] - 0s 16ms/step
3%		23/720 [00:14<03:55, 2.96it/s]1/1 [=====] - 0s 15ms/step
3%		24/720 [00:14<03:53, 2.99it/s]1/1 [=====] - 0s 16ms/step
3%		25/720 [00:14<03:53, 2.98it/s]1/1 [=====] - 0s 16ms/step
4%		26/720 [00:15<03:50, 3.01it/s]1/1 [=====] - 0s 24ms/step
4%		27/720 [00:15<03:52, 2.99it/s]1/1 [=====] - 0s 15ms/step
4%		28/720 [00:15<03:51, 2.99it/s]1/1 [=====] - 0s 16ms/step
4%		29/720 [00:16<03:50, 3.00it/s]1/1 [=====] - 0s 20ms/step
4%		30/720 [00:16<03:52, 2.96it/s]1/1 [=====] - 0s 16ms/step
4%		31/720 [00:16<03:50, 2.98it/s]1/1 [=====] - 0s 15ms/step
4%		32/720 [00:17<03:49, 2.99it/s]1/1 [=====] - 0s 16ms/step
5%		33/720 [00:17<03:49, 3.00it/s]1/1 [=====] - 0s 15ms/step
5%		34/720 [00:17<03:47, 3.02it/s]1/1 [=====] - 0s 15ms/step
5%		35/720 [00:18<03:47, 3.01it/s]1/1 [=====] - 0s 19ms/step
5%		36/720 [00:18<03:46, 3.01it/s]1/1 [=====] - 0s 18ms/step
5%		37/720 [00:18<03:48, 2.99it/s]1/1 [=====] - 0s 15ms/step
5%		38/720 [00:19<03:48, 2.99it/s]1/1 [=====] - 0s 18ms/step
5%		39/720 [00:19<03:50, 2.96it/s]1/1 [=====] - 0s 15ms/step
6%		40/720 [00:19<03:49, 2.97it/s]1/1 [=====] - 0s 16ms/step

Generator Network

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 96, 96, 3)]	0	[]
conv2d (Conv2D)	(None, 96, 96, 64)	15616	['input_1[0][0]']
re_lu (ReLU)	(None, 96, 96, 64)	0	['conv2d[0][0]']
conv2d_2 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu[0][0]']
batch_normalization_1 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_2[0][0]']
re_lu_2 (ReLU)	(None, 96, 96, 64)	0	['batch_normalization_1[0][0]']
conv2d_3 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_2[0][0]']
batch_normalization_2 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_3[0][0]']
add (Add)	(None, 96, 96, 64)	0	['re_lu[0][0]', 'batch_normalization_2[0][0]']
re_lu_3 (ReLU)	(None, 96, 96, 64)	0	['add[0][0]']
conv2d_5 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_3[0][0]']
batch_normalization_4 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_5[0][0]']
re_lu_5 (ReLU)	(None, 96, 96, 64)	0	['batch_normalization_4[0][0]']
conv2d_6 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_5[0][0]']
batch_normalization_5 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_6[0][0]']

add_1 (Add)	(None, 96, 96, 64)	0	['re_lu_3[0][0]', 'batch_normalization_5[0][0]']
re_lu_6 (ReLU)	(None, 96, 96, 64)	0	['add_1[0][0]']
conv2d_8 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_6[0][0]']
batch_normalization_7 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_8[0][0]']
re_lu_8 (ReLU)	(None, 96, 96, 64)	0	['batch_normalization_7[0][0]']
conv2d_9 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_8[0][0]']
batch_normalization_8 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_9[0][0]']
add_2 (Add)	(None, 96, 96, 64)	0	['re_lu_6[0][0]', 'batch_normalization_8[0][0]']
re_lu_9 (ReLU)	(None, 96, 96, 64)	0	['add_2[0][0]']
conv2d_11 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_9[0][0]']
batch_normalization_10 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_11[0][0]']
re_lu_11 (ReLU)	(None, 96, 96, 64)	0	['batch_normalization_10[0][0]']
conv2d_12 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_11[0][0]']
batch_normalization_11 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_12[0][0]']
add_3 (Add)	(None, 96, 96, 64)	0	['re_lu_9[0][0]', 'batch_normalization_11[0][0]']
re_lu_12 (ReLU)	(None, 96, 96, 64)	0	['add_3[0][0]']

conv2d_14 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_12[0][0]']
batch_normalization_13 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_14[0][0]']
re_lu_14 (ReLU)	(None, 96, 96, 64)	0	['batch_normalization_13[0][0]']
conv2d_15 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_14[0][0]']
batch_normalization_14 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_15[0][0]']
add_4 (Add)	(None, 96, 96, 64)	0	['re_lu_12[0][0]', 'batch_normalization_14[0][0]']
re_lu_15 (ReLU)	(None, 96, 96, 64)	0	['add_4[0][0]']
conv2d_17 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_15[0][0]']
batch_normalization_16 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_17[0][0]']
re_lu_17 (ReLU)	(None, 96, 96, 64)	0	['batch_normalization_16[0][0]']
conv2d_18 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_17[0][0]']
batch_normalization_17 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_18[0][0]']
add_5 (Add)	(None, 96, 96, 64)	0	['re_lu_15[0][0]', 'batch_normalization_17[0][0]']
re_lu_18 (ReLU)	(None, 96, 96, 64)	0	['add_5[0][0]']
conv2d_20 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_18[0][0]']
batch_normalization_19 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_20[0][0]']

re_lu_20 (ReLU)	(None, 96, 96, 64)	0	['batch_normalization_19[0][0]']
conv2d_21 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_20[0][0]']
batch_normalization_20 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_21[0][0]']
add_6 (Add)	(None, 96, 96, 64)	0	['re_lu_18[0][0]', 'batch_normalization_20[0][0]']
re_lu_21 (ReLU)	(None, 96, 96, 64)	0	['add_6[0][0]']
conv2d_23 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_21[0][0]']
batch_normalization_22 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_23[0][0]']
re_lu_23 (ReLU)	(None, 96, 96, 64)	0	['batch_normalization_22[0][0]']
conv2d_24 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_23[0][0]']
batch_normalization_23 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_24[0][0]']
add_7 (Add)	(None, 96, 96, 64)	0	['re_lu_21[0][0]', 'batch_normalization_23[0][0]']
re_lu_24 (ReLU)	(None, 96, 96, 64)	0	['add_7[0][0]']
conv2d_26 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_24[0][0]']
batch_normalization_25 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_26[0][0]']
re_lu_26 (ReLU)	(None, 96, 96, 64)	0	['batch_normalization_25[0][0]']
conv2d_27 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_26[0][0]']
batch_normalization_26 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_27[0][0]']

add_8 (Add)	(None, 96, 96, 64)	0	['re_lu_24[0][0]', 'batch_normalization_26[0][0]']
re_lu_27 (ReLU)	(None, 96, 96, 64)	0	['add_8[0][0]']
conv2d_29 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_27[0][0]']
batch_normalization_28 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_29[0][0]']
re_lu_29 (ReLU)	(None, 96, 96, 64)	0	['batch_normalization_28[0][0]']
conv2d_30 (Conv2D)	(None, 96, 96, 64)	36928	['re_lu_29[0][0]']
batch_normalization_29 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_30[0][0]']
add_9 (Add)	(None, 96, 96, 64)	0	['re_lu_27[0][0]', 'batch_normalization_29[0][0]']
re_lu_30 (ReLU)	(None, 96, 96, 64)	0	['add_9[0][0]']
conv2d_32 (Conv2D)	(None, 96, 96, 16)	1040	['re_lu_30[0][0]']
conv2d_34 (Conv2D)	(None, 96, 96, 32)	2080	['re_lu_30[0][0]']
max_pooling2d (MaxPooling2D)	(None, 96, 96, 64)	0	['re_lu_30[0][0]']
conv2d_31 (Conv2D)	(None, 96, 96, 8)	520	['re_lu_30[0][0]']
conv2d_33 (Conv2D)	(None, 96, 96, 16)	2320	['conv2d_32[0][0]']
conv2d_35 (Conv2D)	(None, 96, 96, 32)	25632	['conv2d_34[0][0]']
conv2d_36 (Conv2D)	(None, 96, 96, 64)	4160	['max_pooling2d[0][0]']

concatenate (Concatenate)	(None, 96, 96, 120)	0	['conv2d_31[0][0]', 'conv2d_33[0][0]', 'conv2d_35[0][0]', 'conv2d_36[0][0]']
max_pooling2d_1 (MaxPooling2D)	(None, 96, 96, 120)	0	['concatenate[0][0]']
conv2d_38 (Conv2D)	(None, 96, 96, 16)	1936	['max_pooling2d_1[0][0]']
conv2d_40 (Conv2D)	(None, 96, 96, 32)	3872	['max_pooling2d_1[0][0]']
max_pooling2d_2 (MaxPooling2D)	(None, 96, 96, 120)	0	['max_pooling2d_1[0][0]']
conv2d_37 (Conv2D)	(None, 96, 96, 8)	968	['max_pooling2d_1[0][0]']
conv2d_39 (Conv2D)	(None, 96, 96, 16)	2320	['conv2d_38[0][0]']
conv2d_41 (Conv2D)	(None, 96, 96, 32)	25632	['conv2d_40[0][0]']
conv2d_42 (Conv2D)	(None, 96, 96, 64)	7744	['max_pooling2d_2[0][0]']
concatenate_1 (Concatenate)	(None, 96, 96, 120)	0	['conv2d_37[0][0]', 'conv2d_39[0][0]', 'conv2d_41[0][0]', 'conv2d_42[0][0]']
conv2d_44 (Conv2D)	(None, 96, 96, 16)	1936	['concatenate_1[0][0]']
conv2d_46 (Conv2D)	(None, 96, 96, 32)	3872	['concatenate_1[0][0]']
max_pooling2d_3 (MaxPooling2D)	(None, 96, 96, 120)	0	['concatenate_1[0][0]']
conv2d_43 (Conv2D)	(None, 96, 96, 8)	968	['concatenate_1[0][0]']
conv2d_45 (Conv2D)	(None, 96, 96, 16)	2320	['conv2d_44[0][0]']
conv2d_47 (Conv2D)	(None, 96, 96, 32)	25632	['conv2d_46[0][0]']

conv2d_48 (Conv2D)	(None, 96, 96, 64)	7744	['max_pooling2d_3[0][0]']
concatenate_2 (Concatenate)	(None, 96, 96, 120)	0	['conv2d_43[0][0]', 'conv2d_45[0][0]', 'conv2d_47[0][0]', 'conv2d_48[0][0]']
max_pooling2d_4 (MaxPooling2D)	(None, 96, 96, 120)	0	['concatenate_2[0][0]']
conv2d_50 (Conv2D)	(None, 96, 96, 16)	1936	['max_pooling2d_4[0][0]']
conv2d_52 (Conv2D)	(None, 96, 96, 32)	3872	['max_pooling2d_4[0][0]']
max_pooling2d_5 (MaxPooling2D)	(None, 96, 96, 120)	0	['max_pooling2d_4[0][0]']
conv2d_49 (Conv2D)	(None, 96, 96, 8)	968	['max_pooling2d_4[0][0]']
conv2d_51 (Conv2D)	(None, 96, 96, 16)	2320	['conv2d_50[0][0]']
conv2d_53 (Conv2D)	(None, 96, 96, 32)	25632	['conv2d_52[0][0]']
conv2d_54 (Conv2D)	(None, 96, 96, 64)	7744	['max_pooling2d_5[0][0]']
concatenate_3 (Concatenate)	(None, 96, 96, 120)	0	['conv2d_49[0][0]', 'conv2d_51[0][0]', 'conv2d_53[0][0]', 'conv2d_54[0][0]']
conv2d_55 (Conv2D)	(None, 96, 96, 64)	69184	['concatenate_3[0][0]']
batch_normalization_30 (Batch Normalization)	(None, 96, 96, 64)	256	['conv2d_55[0][0]']
conv2d_56 (Conv2D)	(None, 96, 96, 512)	295424	['batch_normalization_30[0][0]']
up_sampling2d (UpSampling2D)	(None, 192, 192, 512)	0	['conv2d_56[0][0]']
re_lu_31 (ReLU)	(None, 192, 192, 512)	0	['up_sampling2d[0][0]']
conv2d_57 (Conv2D)	(None, 192, 192, 512)	2359808	['re_lu_31[0][0]']
up_sampling2d_1 (UpSampling2D)	(None, 384, 384, 512)	0	['conv2d_57[0][0]']
re_lu_32 (ReLU)	(None, 384, 384, 512)	0	['up_sampling2d_1[0][0]']
conv2d_58 (Conv2D)	(None, 384, 384, 3)	124419	['re_lu_32[0][0]']

=====

Total params: 3,771,555
Trainable params: 3,768,867
Non-trainable params: 2,688

Discriminator Network

Model: "model_1"

Layer (type)	Output Shape	Param #	Connected to
=====			
input_2 (InputLayer)	[(None, 384, 384, 3)]	0	[]
conv2d_59 (Conv2D)	(None, 384, 384, 4)	112	['input_2[0][0]']
leaky_re_lu (LeakyReLU)	(None, 384, 384, 4)	0	['conv2d_59[0][0]']
conv2d_60 (Conv2D)	(None, 192, 192, 4)	148	['leaky_re_lu[0][0]']
batch_normalization_31 (Batch Normalization)	(None, 192, 192, 4)	16	['conv2d_60[0][0]']
leaky_re_lu_1 (LeakyReLU)	(None, 192, 192, 4)	0	['batch_normalization_31[0][0]']
conv2d_61 (Conv2D)	(None, 192, 192, 8)	296	['leaky_re_lu_1[0][0]']
batch_normalization_32 (Batch Normalization)	(None, 192, 192, 8)	32	['conv2d_61[0][0]']
leaky_re_lu_2 (LeakyReLU)	(None, 192, 192, 8)	0	['batch_normalization_32[0][0]']
conv2d_62 (Conv2D)	(None, 96, 96, 8)	584	['leaky_re_lu_2[0][0]']
batch_normalization_33 (Batch Normalization)	(None, 96, 96, 8)	32	['conv2d_62[0][0]']
leaky_re_lu_3 (LeakyReLU)	(None, 96, 96, 8)	0	['batch_normalization_33[0][0]']
conv2d_63 (Conv2D)	(None, 96, 96, 16)	1168	['leaky_re_lu_3[0][0]']
batch_normalization_34 (Batch Normalization)	(None, 96, 96, 16)	64	['conv2d_63[0][0]']
leaky_re_lu_4 (LeakyReLU)	(None, 96, 96, 16)	0	['batch_normalization_34[0][0]']

conv2d_64 (Conv2D)	(None, 48, 48, 16)	2320	['leaky_re_lu_4[0][0]']
batch_normalization_35 (Batch Normalization)	(None, 48, 48, 16)	64	['conv2d_64[0][0]']
leaky_re_lu_5 (LeakyReLU)	(None, 48, 48, 16)	0	['batch_normalization_35[0][0]']
conv2d_66 (Conv2D)	(None, 48, 48, 16)	272	['leaky_re_lu_5[0][0]']
conv2d_68 (Conv2D)	(None, 48, 48, 32)	544	['leaky_re_lu_5[0][0]']
max_pooling2d_6 (MaxPooling2D)	(None, 48, 48, 16)	0	['leaky_re_lu_5[0][0]']
conv2d_65 (Conv2D)	(None, 48, 48, 8)	136	['leaky_re_lu_5[0][0]']
conv2d_67 (Conv2D)	(None, 48, 48, 16)	2320	['conv2d_66[0][0]']
conv2d_69 (Conv2D)	(None, 48, 48, 32)	25632	['conv2d_68[0][0]']
conv2d_70 (Conv2D)	(None, 48, 48, 64)	1088	['max_pooling2d_6[0][0]']
concatenate_4 (Concatenate)	(None, 48, 48, 120)	0	['conv2d_65[0][0]', 'conv2d_67[0][0]', 'conv2d_69[0][0]', 'conv2d_70[0][0]']
max_pooling2d_7 (MaxPooling2D)	(None, 48, 48, 120)	0	['concatenate_4[0][0]']
conv2d_72 (Conv2D)	(None, 48, 48, 16)	1936	['max_pooling2d_7[0][0]']
conv2d_74 (Conv2D)	(None, 48, 48, 32)	3872	['max_pooling2d_7[0][0]']
max_pooling2d_8 (MaxPooling2D)	(None, 48, 48, 120)	0	['max_pooling2d_7[0][0]']
conv2d_71 (Conv2D)	(None, 48, 48, 8)	968	['max_pooling2d_7[0][0]']
conv2d_73 (Conv2D)	(None, 48, 48, 16)	2320	['conv2d_72[0][0]']

conv2d_75 (Conv2D)	(None, 48, 48, 32)	25632	['conv2d_74[0][0]']
conv2d_76 (Conv2D)	(None, 48, 48, 64)	7744	['max_pooling2d_8[0][0]']
concatenate_5 (Concatenate)	(None, 48, 48, 120)	0	['conv2d_71[0][0]', 'conv2d_73[0][0]', 'conv2d_75[0][0]', 'conv2d_76[0][0]']
conv2d_78 (Conv2D)	(None, 48, 48, 16)	1936	['concatenate_5[0][0]']
conv2d_80 (Conv2D)	(None, 48, 48, 32)	3872	['concatenate_5[0][0]']
max_pooling2d_9 (MaxPooling2D)	(None, 48, 48, 120)	0	['concatenate_5[0][0]']
conv2d_77 (Conv2D)	(None, 48, 48, 8)	968	['concatenate_5[0][0]']
conv2d_79 (Conv2D)	(None, 48, 48, 16)	2320	['conv2d_78[0][0]']
conv2d_81 (Conv2D)	(None, 48, 48, 32)	25632	['conv2d_80[0][0]']
conv2d_82 (Conv2D)	(None, 48, 48, 64)	7744	['max_pooling2d_9[0][0]']
concatenate_6 (Concatenate)	(None, 48, 48, 120)	0	['conv2d_77[0][0]', 'conv2d_79[0][0]', 'conv2d_81[0][0]', 'conv2d_82[0][0]']
max_pooling2d_10 (MaxPooling2D)	(None, 48, 48, 120)	0	['concatenate_6[0][0]']
conv2d_84 (Conv2D)	(None, 48, 48, 16)	1936	['max_pooling2d_10[0][0]']
conv2d_86 (Conv2D)	(None, 48, 48, 32)	3872	['max_pooling2d_10[0][0]']
max_pooling2d_11 (MaxPooling2D)	(None, 48, 48, 120)	0	['max_pooling2d_10[0][0]']

conv2d_83 (Conv2D)	(None, 48, 48, 8)	968	['max_pooling2d_10[0][0]']
conv2d_85 (Conv2D)	(None, 48, 48, 16)	2320	['conv2d_84[0][0]']
conv2d_87 (Conv2D)	(None, 48, 48, 32)	25632	['conv2d_86[0][0]']
conv2d_88 (Conv2D)	(None, 48, 48, 64)	7744	['max_pooling2d_11[0][0]']
concatenate_7 (Concatenate)	(None, 48, 48, 120)	0	['conv2d_83[0][0]', 'conv2d_85[0][0]', 'conv2d_87[0][0]', 'conv2d_88[0][0]']
conv2d_89 (Conv2D)	(None, 48, 48, 32)	34592	['concatenate_7[0][0]']
batch_normalization_36 (Batch Normalization)	(None, 48, 48, 32)	128	['conv2d_89[0][0]']
leaky_re_lu_6 (LeakyReLU)	(None, 48, 48, 32)	0	['batch_normalization_36[0][0]']
conv2d_90 (Conv2D)	(None, 24, 24, 32)	9248	['leaky_re_lu_6[0][0]']
batch_normalization_37 (Batch Normalization)	(None, 24, 24, 32)	128	['conv2d_90[0][0]']
leaky_re_lu_7 (LeakyReLU)	(None, 24, 24, 32)	0	['batch_normalization_37[0][0]']
flatten (Flatten)	(None, 18432)	0	['leaky_re_lu_7[0][0]']
dense (Dense)	(None, 64)	1179712	['flatten[0][0]']
leaky_re_lu_8 (LeakyReLU)	(None, 64)	0	['dense[0][0]']
dense_1 (Dense)	(None, 1)	65	['leaky_re_lu_8[0][0]']
=====			
Total params: 1,386,117			
Trainable params: 1,385,885			
Non-trainable params: 232			

VGG16

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16\_weights\_tf\_dim\_ordering\_tf\_kernels\_notop.h5
58889256/58889256 [=====] - 2s 0us/step
Model: "model_2"
```

Layer (type)	Output Shape	Param #
input_3 (InputLayer)	(None, 384, 384, 3)	0
block1_conv1 (Conv2D)	(None, 384, 384, 64)	1792
block1_conv2 (Conv2D)	(None, 384, 384, 64)	36928
block1_pool (MaxPooling2D)	(None, 192, 192, 64)	0
block2_conv1 (Conv2D)	(None, 192, 192, 128)	73856
block2_conv2 (Conv2D)	(None, 192, 192, 128)	147584
block2_pool (MaxPooling2D)	(None, 96, 96, 128)	0
block3_conv1 (Conv2D)	(None, 96, 96, 256)	295168
block3_conv2 (Conv2D)	(None, 96, 96, 256)	590080
block3_conv3 (Conv2D)	(None, 96, 96, 256)	590080
block3_pool (MaxPooling2D)	(None, 48, 48, 256)	0

```
=====
Total params: 1,735,488
Trainable params: 1,735,488
Non-trainable params: 0
```

Combined Model

Model: "model_3"

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 96, 96, 3)]	0	[]
model (Functional)	(None, 384, 384, 3)	3771555	['input_1[0][0]']
input_2 (InputLayer)	[(None, 384, 384, 3)]	0	[]
model_1 (Functional)	(None, 1)	1386117	['model[0][0]']
model_2 (Functional)	(None, 48, 48, 256)	1735488	['model[0][0]']

Total params: 6,893,160

Trainable params: 3,768,867

Non-trainable params: 3,124,293

ANNEXURE 4

GitHub link

<https://github.com/Ashima-Jain2001/GAN-Inception-Residual-SR>