****

**REPORT**

**TOPIC:**

**Enhancement Of Low-resolution Images to High-Resolution Images Using Generative Adversarial Networks**

Submitted By:

[UDUTA SRI MAHA CHANAKYA CHANDRA](mailto:21bsm059@iiitdmj.ac.in)

21BSM059

PDPM Indian Institute of Information Technology, Design and Manufacturing, Jabalpur,482005

**CONTENTS**

1. Introduction
   1. Artificial Intelligence and Machine Learning
2. Neural Networks
   1. Introduction to Neural Networks
   2. Convolutional Neural Networks

2.2.1 Inside convolutional neural networks

2.2.2 CNN Layers

1. GAN Neural Networks
2. TRADITIONAL IMAGE SUPER RESOLUTION
3. SR GAN’s
4. Appendix

6.1 Importing the Essential Libraries

6.2 Generator Model

6.3 Discriminator Model

6.4 Defining the Loss Functions

6.5 Training the GAN model with the prepared dataset

# Result

7.1 Checking Confidence with YOLOv4Conclusion

# Conclusion

# Reference

# INTRODUCTION

Artificial Intelligence (AI) and Machine Learning (ML) are transformative technologies that have revolutionized numerous industries and are driving

innovation in today's digital age. This document provides a concise overview of AI and ML, explaining their fundamental concepts and applications.

**Artificial Intelligence (AI)**:

AI refers to the simulation of human intelligence in machines that are programmed to think and learn like humans. It involves the development of computer systems capable of performing tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and problem-solving.

AI can be categorized into two types: Narrow AI and General AI. Narrow AI, also known as Weak AI, is designed to perform specific tasks and excel in narrow domains. General AI, on the other hand, aims to exhibit human-like intelligence across a wide range of tasks and domains.

AI encompasses various subfields, including Natural Language Processing (NLP), Computer Vision, Robotics, Expert Systems, and Neural Networks.

These subfields utilize different techniques and algorithms to achieve specific AI functionalities.

**Machine Learning (ML):**

Machine Learning is a subset of AI that focuses on the development of algorithms and statistical models that enable computer systems to learn from data and make predictions or decisions without being explicitly programmed. ML algorithms learn patterns and relationships from the data and use that knowledge to perform tasks or make predictions.

ML can be categorized into three types: Supervised Learning, Unsupervised Learning, and Reinforcement Learning. Supervised Learning involves training ML models using labelled data.

Unsupervised Learning, on the other hand, deals with unlabelled data and aims to discover hidden patterns or structures within the data. Reinforcement Learning involves training models through an interactive process of trial and error, where the model learns from feedback or rewards received based on its actions.

ML algorithms include Decision Trees, Support Vector Machines, Naive Bayes, Neural Networks, and more. These algorithms are used for various tasks, such as classification, regression, clustering, and recommendation systems.

**Applications of AI and ML:**

AI and ML have found applications across numerous industries, transforming the way businesses operate and improving various aspects of our lives. Some key applications include:

* Healthcare: AI and ML are used for disease diagnosis, drug discovery, personalized medicine, patient monitoring, and healthcare management systems.
* Finance: AI and ML algorithms are used for fraud detection, risk assessment, algorithmic trading, credit scoring, and customer service automation.
* Transportation: AI and ML technologies enable autonomous vehicles, traffic optimization, predictive maintenance, and route planning.
* Retail: AI and ML are used for personalized marketing, demand

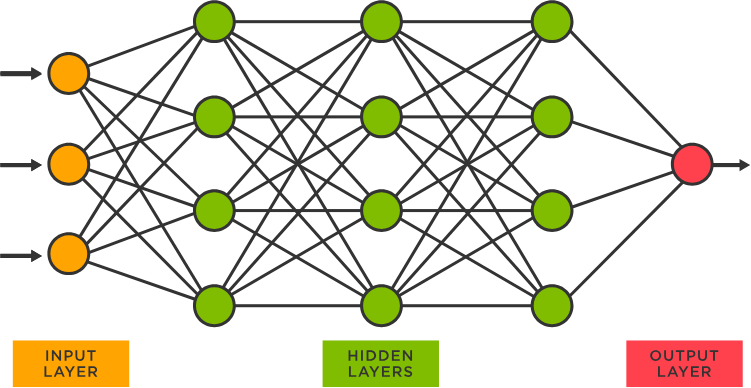
forecasting, inventory management, and recommendation systems.

* Education: AI and ML are applied in adaptive learning systems, intelligent tutoring, plagiarism detection, and educational data analysis.
* Manufacturing: AI and ML are used for predictive maintenance, quality control, supply chain optimization, and robotics automation.
* Defense: AI in defense enables autonomous systems for surveillance and combat, enhances threat detection and analysis capabilities, and strengthens cybersecurity measures against evolving cyber threats. It also provides decision support systems and aids in intelligence gathering and analysis for enhanced situational awa

**2 NEURAL NETWORKS  
2.1 INTRODUCTION**

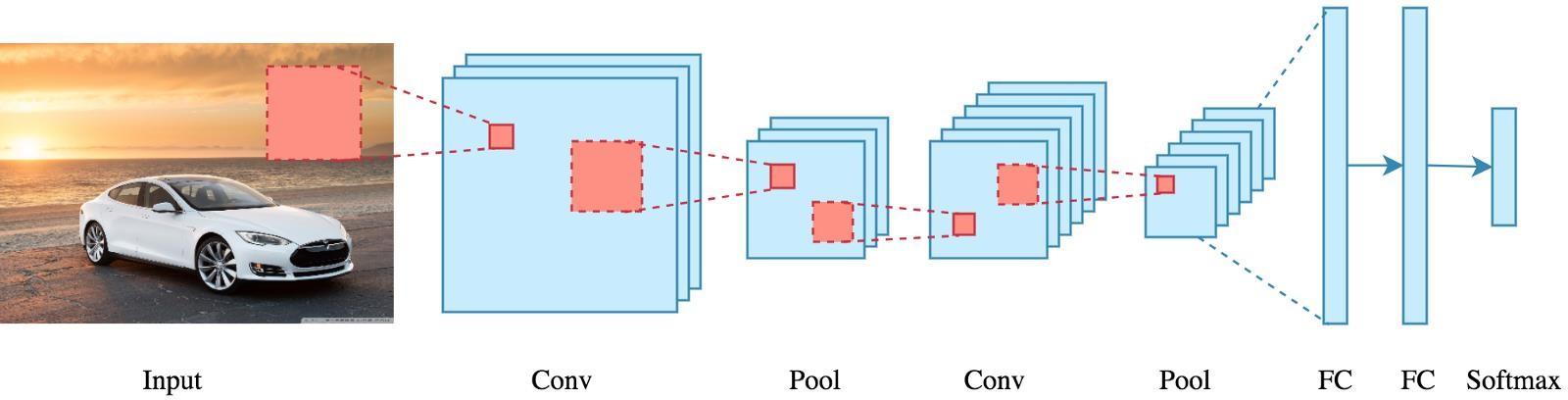
A neural network is an interconnected assembly of simple processing elements units or nodes whose functionality is loosely based on animal neurons. In simple words, the mechanism of neural networks mimics that of a human or animal brain. The processing ability of the network will be stored in the inter-unit connection strengths, or weights, obtained by a process of adaptation to or learning from a set of training patterns.

Neural networks and deep learning are big topics in Computer Science and in the technology industry, they currently provide the best solutions to many problems in image recognition, speech recognition, and natural language processing. Recently many papers have been published featuring AI that can learn to paint, build 3D Models, and create user interfaces(pix2code), some create images given a sentence and there are many more incredible things being done every day using neural networks.

It is this architecture and style of processing of a biological neuron, that we hope to incorporate in neural networks and, because of the emphasis on the importance of the interneuronConclusion connections, this type of system is sometimes referred to as being connectionist and the study of this general approach as connectionism. This terminology is often the one encountered for neural networks in the context of psychologically inspired models of human cognitive function. However, we will use it quite generally to refer to neural networks without reference to any particular field of application.

## 2.2 Convolutional Neural Networks

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.



#### 2.2.1 Inside convolutional neural networks

Artificial neural networks (ANNs) are a core element of deep learning algorithms. One type of ANN is a recurrent neural network (RNN) that uses sequential or time series data as input. It is suitable for applications involving natural language processing (NLP), language translation, speech recognition, and image captioning.

CNN is another type of neural network that can uncover key information in both time series and image data. For this reason, it is highly valuable for image-related tasks, such as image recognition, object classification and pattern recognition. To identify patterns within an image, a CNN leverages principles from linear algebra, such as matrix multiplication. CNNs can also classify audio and signal data.

A CNN's architecture is analogous to the connectivity pattern of the human brain. Just like the brain consists of billions of neurons, CNNs also have neurons arranged in a specific way. In fact, a CNN's neurons are arranged like the brain's

frontal lobe, the area responsible for processing visual stimuli. This arrangement ensures that the entire visual field is covered, thus avoiding the piecemeal image processing problem of traditional neural networks, which must be fed images in reduced-resolution pieces. Compared to the older networks, a CNN delivers better performance with image inputs, and also with speech or audio signal inputs

#### 2.2.2 CNN layers

A deep-learning CNN consists of three layers: a convolutional layer, a pooling layer, and a fully connected (FC) layer. The convolutional layer is the first layer while the FC layer is the last.

From the convolutional layer to the FC layer, the complexity of the CNN increases. It is this increasing complexity that allows the CNN to successfully identify larger portions and more complex features of an image until it finally identifies the object in its entirety.

Convolutional layer. The majority of computations happen in the convolutional layer, which is the core building block of a CNN. A second convolutional layer can follow the initial convolutional layer. The process of convolution involves a kernel or filter inside this layer moving across the receptive fields of the image, checking if a feature is present in the image.

Over multiple iterations, the kernel sweeps over the entire image. After each iteration, a dot product is calculated between the input pixels and the filter. The final output from the series of dots is known as a feature map or convolved feature. Ultimately, the image is converted into numerical values in this layer, allowing the CNN to interpret the image and extract relevant patterns.

Pooling layer. Like the convolutional layer, the pooling layer also sweeps a kernel or filter across the input image. But unlike the convolutional layer, the pooling layer reduces the number of parameters in the input and also results in some information loss. On the positive side, this layer reduces complexity and improves the efficiency of the CNN.

Fully connected layer. The FC layer is where image classification happens in the CNN based on the features extracted in the previous layers. Here, fully connected means that all the inputs or nodes from one layer are connected to every activation unit or node of the next layer.

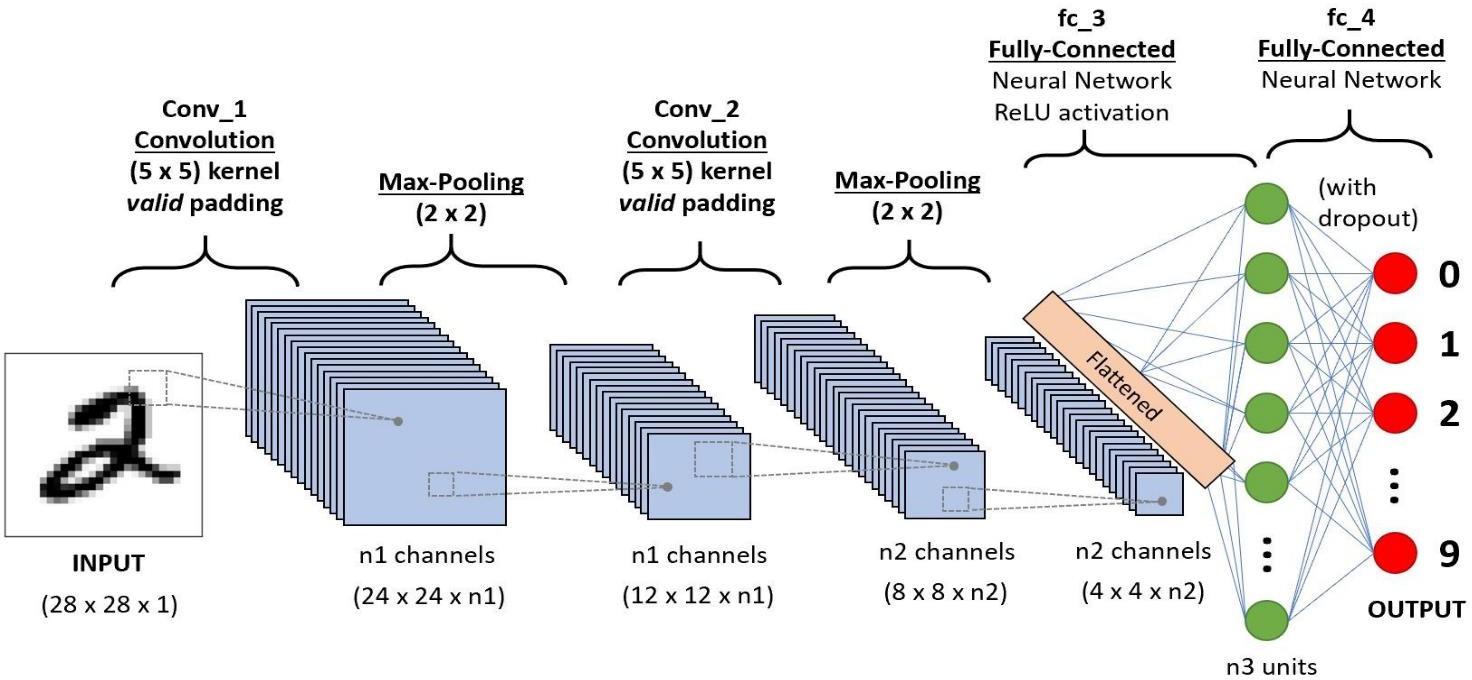
All the layers in the CNN are not fully connected because it would result in an unnecessarily dense network. It also would increase losses and affect the output quality, and it would be computationally expensive.

How do convolutional neural networks work?

A CNN can have multiple layers, each of which learns to detect the different features of an input image. A filter or kernel is applied to each image to produce an output that gets progressively better and more detailed after each layer. In the lower layers, the filters can start as simple features.

At each successive layer, the filters increase in complexity to check and identify features that uniquely represent the input object. Thus, the output of each convolved image -- the partially recognized image after each layer -- becomes the input for the next layer. In the last layer, which is an FC layer, the CNN recognizes the image or the object it represents.

With convolution, the input image goes through a set of these filters. As each filter activates certain features from the image, it does its work and passes on its output to the filter in the next layer. Finally, all the image data progressing through the CNN's multiple layers allow the CNN to identify the entire object.



CNNs vs. Neural Networks

The biggest problem with regular neural networks (NNs) is a lack of scalability. For smaller images with fewer color channels, a regular NN may produce satisfactory results. But as the size and complexity of an image increases, the need for computational power and resources also increases which necessitates a larger and more expensive NN.

Moreover, the problem of overfitting also arises over time, wherein the NN tries to learn too many details in the training data. It may also end up learning the noise in the data, which affects its performance on test data sets. Ultimately, the NN fails to identify the features or patterns in the data set and thus the object itself.

**3. Generative Adversarial Network**

A Generative Adversarial Network (GAN) is a powerful neural network architecture that has revolutionized the field of generative modeling. GANs have gained significant attention due to their ability to generate realistic and high-quality synthetic data, such as images, audio, and text. In this description, we will delve into the inner workings of GANs and their training process.

A generator network and a discriminator network are the two fundamental parts of a GAN. The generator attempts to produce samples that resemble the training data it has been exposed to by using random noise as input. The generator first creates random, subpar samples. However, it gains the ability to produce more accurate samples that accurately reflect the underlying distribution of the training data through an iterative training process.

The discriminator, on the other hand, functions as a binary classifier. Its function is to tell authentic samples from the training data apart from phony samples produced by the generator. Using both genuine and false samples, the discriminator is trained to accurately identify them. The discriminator's goal is to become adept at telling authentic samples from false ones.

The adversarial training strategy is the main concept underpinning GANs. In a game of competition, the discriminator and generator are trained concurrently. The discriminator seeks to grow more adept at differentiating between actual and false samples, while the generator aims to create samples that can trick the discriminator into classifying them as real. The generator gains knowledge from the discriminator's feedback to enhance its output, and the discriminator changes to improve its classification accuracy as a result of this adversarial process.

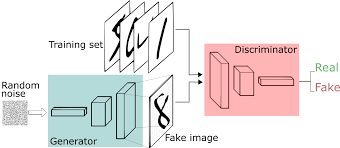
During training, the generator and discriminator update their weights using gradient descent optimization. The generator tries to minimize the probability of the discriminator correctly classifying the generated samples as fake, while the discriminator aims to maximize its ability to correctly classify real and fake samples. This min-max game formulation leads to a Nash equilibrium, where the generator produces highly realistic samples that can fool the discriminator.

One of the challenges in GAN training is achieving a balance between the generator and discriminator. If the generator becomes too powerful, it may generate samples that closely resemble the real data, making it difficult for the discriminator to differentiate between them. On the other hand, if the discriminator becomes too strong, it can easily classify the generated samples as fake, providing little feedback for the generator to improve. Striking the right balance between the two networks is crucial for the success of the GAN.

In addition to the adversarial loss used to train GANs, researchers have introduced various techniques to enhance the training stability and quality of the generated samples. One such technique is the use of auxiliary losses, such as feature matching and gradient penalty, which provide additional training signals to guide the generator and discriminator. Another approach is the incorporation of conditioning information, where both the generator and discriminator are conditioned on additional input, enabling more fine-grained control over the generated samples.

GANs have been applied to numerous domains and have achieved remarkable results. In image synthesis, GANs can generate photo-realistic images that are indistinguishable from real ones. GANs have also been used for style transfer, where the style of one image is transferred to another, and for image-to-image translation tasks, such as converting sketches to realistic images or transforming day-time images into night-time scenes.

Furthermore, GANs have shown promise in natural language processing tasks. Text-based GANs can generate coherent and contextually relevant sentences, paragraphs, and even entire articles. They have also been employed in data augmentation, where GANs can generate synthetic data to augment training sets, improving the performance of models on various tasks.



**4 ￼TRADITIONAL IMAGE SUPER RESOLUTION:**

Image super-resolution is a technique that aims to enhance the quality and resolution of low-resolution images, allowing for greater visual clarity and detail. One of the key components of image super-resolution is image interpolation, which plays a crucial role in reconstructing high-resolution images from their low-resolution counterparts.

Image interpolation refers to the process of estimating the missing pixel values in an image by utilizing the known pixel values. In the context of image super-resolution, image interpolation is used to generate high-resolution images by extrapolating the available low-resolution data.

There are various interpolation methods used in image super-resolution, each with its own strengths and limitations. In this article, we will explore some of the commonly used interpolation techniques and their application in image super-resolution.

**4.1** **Nearest Neighbor Interpolation:**

Nearest Neighbor interpolation is the simplest interpolation method, where each missing pixel value is estimated by assigning the value of its nearest neighboring pixel. While this method is computationally efficient, it often leads to blocky and jagged artifacts due to its inability to capture the smooth transitions in the image.

**4.2** **Bilinear Interpolation:**

Bilinear interpolation estimates missing pixel values by taking a weighted average of the four nearest neighboring pixels. This method considers the distance between the known pixels and the target pixel to calculate the weights. Bilinear interpolation produces smoother results compared to nearest neighbor interpolation, but it may still introduce blurring and loss of sharpness.

**4.3** **Bicubic Interpolation:**

Bicubic interpolation is an extension of bilinear interpolation that incorporates more neighboring pixels for estimating missing values. It uses a 4x4 pixel neighborhood and applies a cubic function to determine the weights. Bicubic interpolation can preserve sharper edges and details compared to bilinear interpolation but may introduce ringing artifacts around high-contrast regions.

**4.4**  **Super-Resolution Specific Interpolation Techniques**:

In addition to traditional interpolation methods, there are specialized techniques developed specifically for image super-resolution.

Deep Learning-based Interpolation: With the advancements in deep learning, convolutional neural networks (CNNs) have been employed for image interpolation in super-resolution. These models are trained on large datasets to learn the mapping between low-resolution and high-resolution images, enabling them to generate high-quality and realistic super-resolved images.

Image interpolation is a crucial component of image super-resolution, allowing for the reconstruction of high-resolution images from low-resolution data. Various interpolation techniques, ranging from simple approaches like nearest neighbor and bilinear interpolation to advanced methods like Lanczos interpolation and deep learning-based approaches, contribute to enhancing the quality and detail of super-resolved images. The choice of interpolation method depends on the specific requirements of the application and the trade-off between computational complexity and visual fidelity. Continued research and development in image interpolation techniques will further advance the field of image super-resolution and improve the quality of reconstructed high-resolution images.

**5 Super Resolution GAN (SRGAN)**

Super Resolution GAN (SRGAN) is a specific application of Generative Adversarial Networks (GANs) designed for the task of image super-resolution. Image super-resolution refers to the process of enhancing the resolution and quality of a low-resolution image to a higher-resolution version.

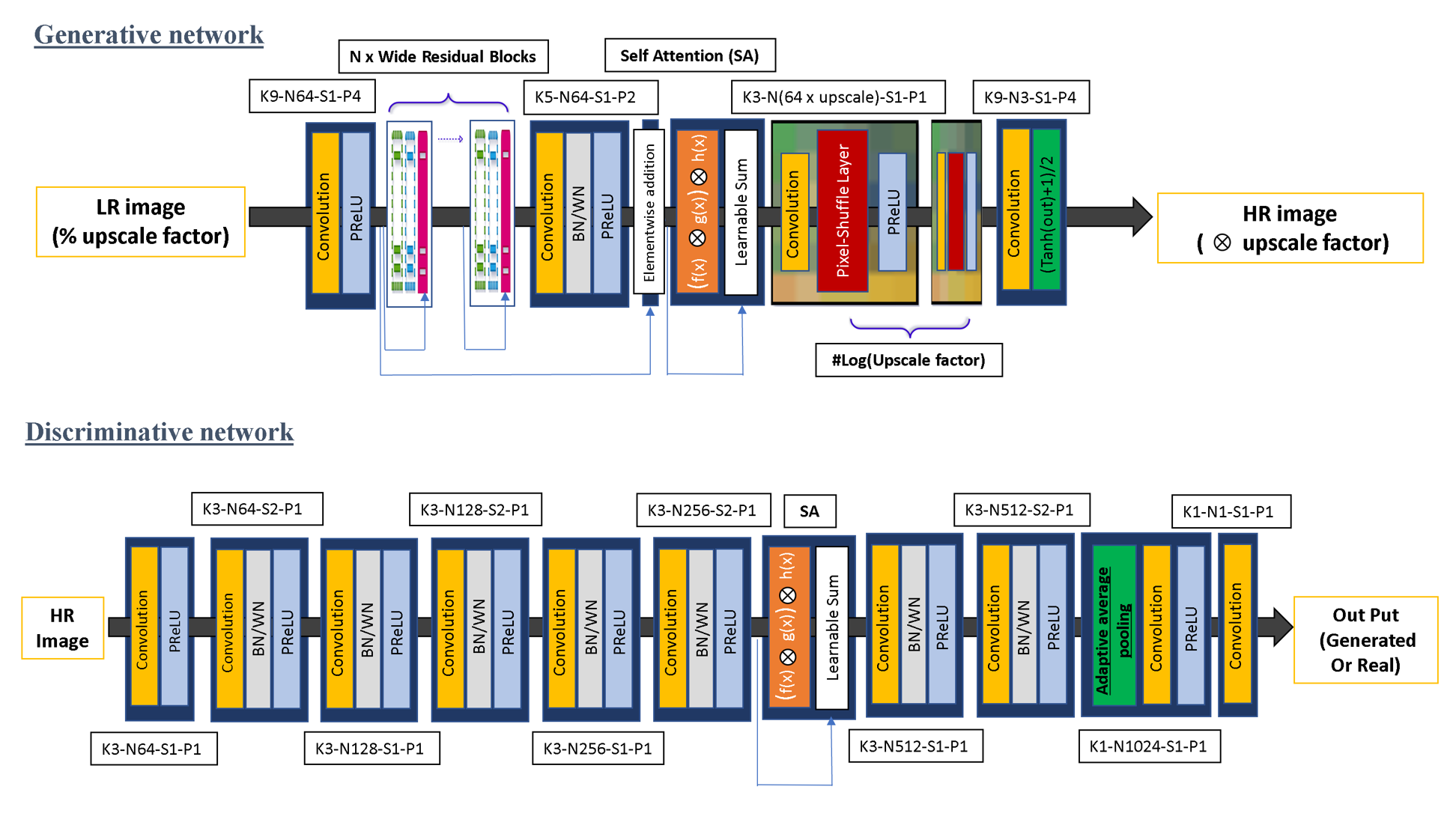
SRGAN consists of a generator network and a discriminator network, similar to the traditional GAN architecture. However, it incorporates additional components to address the super-resolution task specifically.

The generator in SRGAN takes a low-resolution image as input and aims to generate a high-resolution image. It employs a deep convolutional neural network (CNN) to learn the mapping between the low-resolution and high-resolution image spaces. The generator network typically consists of multiple convolutional layers, followed by upsampling layers, which help to increase the resolution of the image.

The discriminator in SRGAN is responsible for distinguishing between the generated high-resolution images from the generator and the real high-resolution images from the training dataset. It uses a CNN-based classifier to classify the input images as real or fake. The discriminator's objective is to correctly classify the real images as real and the generated images as fake.

The training process of SRGAN involves an adversarial training scheme. The generator and discriminator networks are trained in a competitive manner, where the generator tries to produce high-resolution images that can deceive the discriminator, while the discriminator learns to become better at distinguishing between real and generated images. This adversarial training process helps the generator to generate more realistic and visually appealing high-resolution images over time.

To further enhance the perceptual quality of the generated images, SRGAN introduces perceptual loss. Perceptual loss measures the difference between the high-resolution images generated by the generator and the real high-resolution images using pre-trained deep neural networks, such as the VGG network. This loss term encourages the generator to produce images that not only look visually similar to the real images but also capture their high-level features and textures.



**6 APPENDIX**

## 6.1 Importing the Essential libraries

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.image as mpg

import cv2

import os

import tensorflow as tf

import zipfile

from tensorflow.keras.layers import Conv2D,Conv2DTranspose, Dense, Flatten, Input,BatchNormalization,PReLU,LeakyReLU,Concatenate

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.models import Model

from tensorflow.nn import depth\_to\_space

from tensorflow.keras.utils import plot\_model

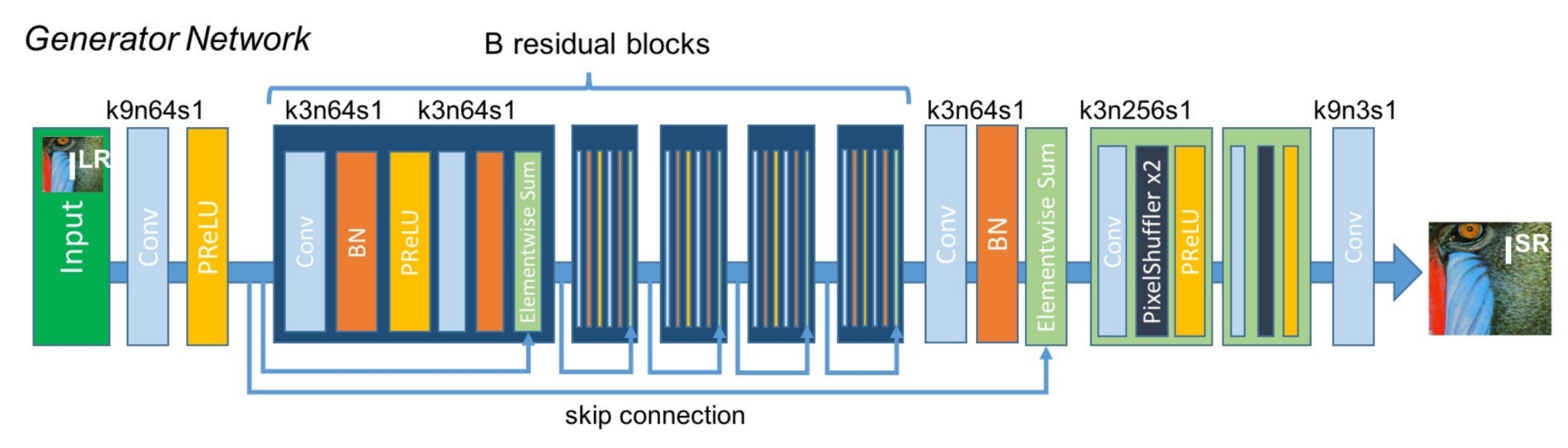
from tensorflow.keras.applications.vgg19 import VGG19

from tensorflow.math import reduce\_mean,square

from tensorflow.keras.losses import MeanSquaredError,BinaryCrossentropy

from tensorflow.keras import Sequential

**6.2.2 Generator Model :**

****

def residual\_blocks(input):

x = Conv2D(64,(3,3),padding='same')(input)

x = BatchNormalization()(x)

x = PReLU()(x)

x = Conv2D(64,(3,3),padding='same')(x)

x = BatchNormalization()(x)

x = Concatenate()([input,x])

return x

def subpixel(input):

x = Conv2D(256\*2,(3,3),padding='same')(input)

x = depth\_to\_space(x,2)

x = PReLU()(x)

return x

def generator():

inputs = Input(shape=(96,96,3))

x1 = Conv2D(64,(9,9),padding='same')(inputs)

x1 = PReLU()(x1)

r = residual\_blocks(x1)

for \_ in range(15):

r = residual\_blocks(r)

x2 = Conv2D(64,(3,3),padding='same')(r)

x2 = BatchNormalization()(x2)

merge = Concatenate()([x2,x1])img = cv2.resize(img,(96,96))

x = subpixel(merge)

x = subpixel(x)

output = Conv2D(3,(9,9),padding='same',activation='tanh')(x)

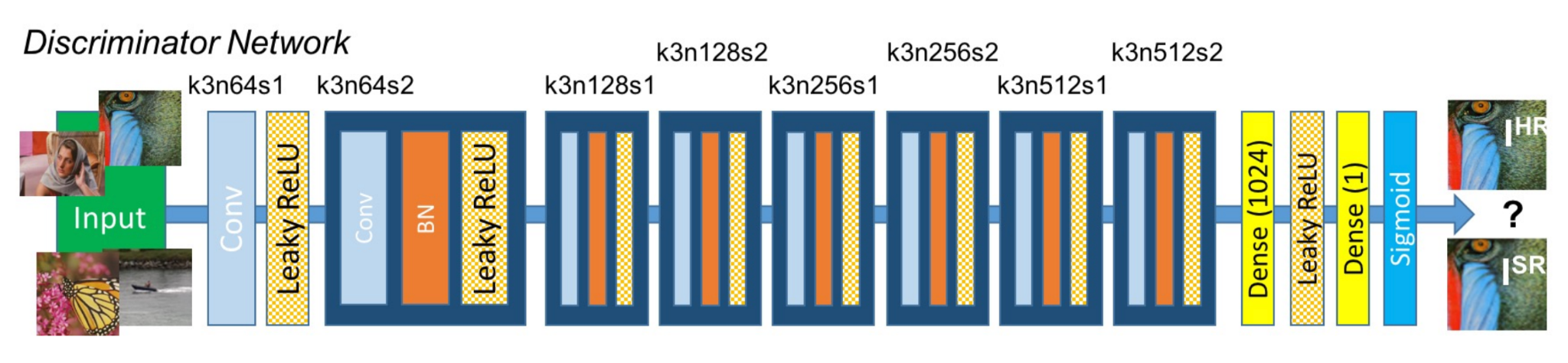
model = Model(inputs,output)

return model

g\_model = generator()

g\_model.summary()

**6.3 Discriminator Model :**

****

def conv\_block(input,n\_channels):

x = Conv2D(n\_channels,(3,3),padding='same')(input)

x = BatchNormalization()(x)

x = LeakyReLU(0.2)(x)

x = Conv2D(n\_channels,(3,3),padding='same',strides=(2,2))(x)

x = BatchNormalization()(x)

x = LeakyReLU(0.2)(x)

return x

def discriminator():

input = Input(shape=(384,384,3))

c1 = Conv2D(64,(3,3),padding='same')(input)

c1 = LeakyReLU(0.2)(c1)

c1 = Conv2D(64,(3,3),padding='same',strides=(2,2))(c1)

c1 = BatchNormalization()(c1)

c1 = LeakyReLU(0.2)(c1)

c2 = conv\_block(c1,128)

c3 = conv\_block(c2,256)

c4 = conv\_block(c3,512)

x = Flatten()(c4)

x = Dense(1024)(x)

x = LeakyReLU(0.2)(x)

output = Dense(1,activation='sigmoid')(x)

model = Model([input],output)

return model

## 6.4 Defining the loss Functions:

1. **Generaor\_loss = Perceptual\_loss ( vgg\_loss ) + Adeversarial\_loss**
2. **Discriminator\_loss = Adversarial\_loss ( Binary\_crossEntropy )**

def vgg\_model():

selected\_layers = ['block3\_conv4' ]

vgg\_model = VGG19(include\_top=False,

input\_shape=(384,384,3))

for layers in vgg\_model.layers:

layers.trainable = False

model = Model(vgg\_model.input,outputs=[vgg\_model.get\_layer(layer).output for layer in selected\_layers])

model.trainable = False

return model

vgg\_model = vgg\_model()

vgg\_model.summary()

vgg\_model.trainable

Downloading data from<https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.h5>

80134624/80134624 [==============================] - 6s 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\_conv4 (Conv2D) (None, 96, 96, 256) 590080

=================================================================

Total params: 2,325,568 , Trainable params: 0 , Non-trainable params: 2,325,560

# 6.5 Training the GAN model with the prepared dataset :

from tensorflow import GradientTape,function

generator\_optimizer = Adam(learning\_rate=0.0001,beta\_1=0.9)

discriminator\_optimizer = Adam(learning\_rate=0.0001,beta\_1=0.9)

@tf.function

def train\_step(lr\_images,hr\_images,n\_samples):

with GradientTape() as disc\_tape:

generated\_images = g\_model(lr\_images,training=True)

real\_output = d\_model(hr\_images,training=True)

fake\_output = d\_model(generated\_images,training=True)

d\_loss = discriminator\_loss(real\_output,fake\_output,n\_samples)

gradients\_of\_discriminator = disc\_tape.gradient(d\_loss, d\_model.trainable\_variables)

discriminator\_optimizer.apply\_gradients(zip(gradients\_of\_discriminator, d\_model.trainable\_variables))

with tf.GradientTape() as gen\_tape:

fake\_output = g\_model(lr\_images)

g\_loss = generator\_loss(fake\_output, hr\_images,n\_samples)

gradients\_of\_generator = gen\_tape.gradient(g\_loss, g\_model.trainable\_variables)

generator\_optimizer.apply\_gradients(zip(gradients\_of\_generator, g\_model.trainable\_variables))

return d\_loss,g\_loss

def train(lr\_images,hr\_images,epochs=50,n\_samples=5):

samples = np.array([1,2,3,4,5])

for epoch in range(epochs):

for batch in range(int(4128//5)-1):

input\_lr = lr[samples]

input\_hr = hr[samples]

samples+=5

d\_loss,g\_loss = train\_step(input\_lr,input\_hr,n\_samples)

samples+=1

print(f'Epoch: {epoch+1} - {batch+1}/{int(4128//5)} - - {d\_loss},{g\_loss}')

if epoch%10 == 0:

file\_name = 'generator\_model'+str(epoch)+'.h5'

g\_model.save(file\_name)

train(lr,hr)

# Epoch: 1 - 650/825 - - 5.439895289782726e-07,28.301271145153734

# Epoch: 1 - 651/825 - - 6.257090422664173e-35,22.672215237265238

# Epoch: 1 - 652/825 - - 3.791878060985087e-24,34.058466764499485

# Epoch: 1 - 653/825 - - 8.670800995334415e-17,38.72041385048456

# Epoch: 1 - 654/825 - - 3.7305512953951514e-14,36.95762221160246

# Epoch: 1 - 655/825 - - 0.00043219391955062747,30.03382506548126

# Epoch: 1 - 656/825 - - 0.18446943163871765,22.799959182739258

# Epoch: 1 - 657/825 - - 1.4327322336342923e-34,34.732643127441406

# Epoch: 1 - 658/825 - - 4.494227409362793,27.598312377929688

# Epoch: 1 - 659/825 - - 5.639137516455029e-15,24.838186264038086

# Epoch: 1 - 660/825 - - 0.8423712849617004,21.15981945624684

# Epoch: 1 - 661/825 - - 1.2057462834986338e-25,16.072620391845703

# Epoch: 1 - 662/825 - - 4.425756742421072e-06,23.530563977210367

# Epoch: 1 - 663/825 - - 2.1339657306671143,34.08419468940315

# Epoch: 1 - 664/825 - - 3.7521121501922607,30.274691464965176

# Epoch: 1 - 665/825 - - 8.499777964798838e-11,29.25064096788403

# Epoch: 1 - 666/825 - - 5.732959270477295,22.707238339850008

# Epoch: 1 - 667/825 - - 8.688492300652112e-22,27.528656005859375

# Epoch: 1 - 668/825 - - 4.665359020233154,38.80454635620117

# Epoch: 1 - 669/825 - - 1.7278072834014893,24.40953270003595

# Epoch: 1 - 670/825 - - 5.054041862487793,35.438994260934365

# Epoch: 1 - 671/825 - - 7.670802663012424e-22,45.912395020335005

# Epoch: 1 - 672/825 - - 0.8065133094787598,31.40310936983208

# Epoch: 1 - 673/825 - - 3.49983286857605,37.55192184448242

# Epoch: 1 - 674/825 - - 3.012951250525118e-16,34.06690979003906

# Epoch: 1 - 675/825 - - 1.0539165517010885e-34,33.639896392822266

# Epoch: 1 - 676/825 - - 5.888573628709537e-09,43.41344451904297

# Epoch: 1 - 677/825 - - 2.4337582296696247e-11,44.97238036215194

# Epoch: 1 - 678/825 - - 0.20365393161773682,20.64662742614746

# Epoch: 1 - 679/825 - - 0.0,21.28672218322754

# Epoch: 1 - 680/825 - - 0.09989918768405914,44.59115982055664

￼￼

# 7 Result:

The results obtained from the image super-resolution process have demonstrated significant improvements in image quality and resolution. The project's effectiveness is evident in the visually enhanced images, which reveal finer details and enhance the visibility of critical elements. By successfully restoring important information from low-quality images, police officers can better identify individuals, objects, and contextual details, thus aiding investigations and potentially leading to quicker resolutions.

# 

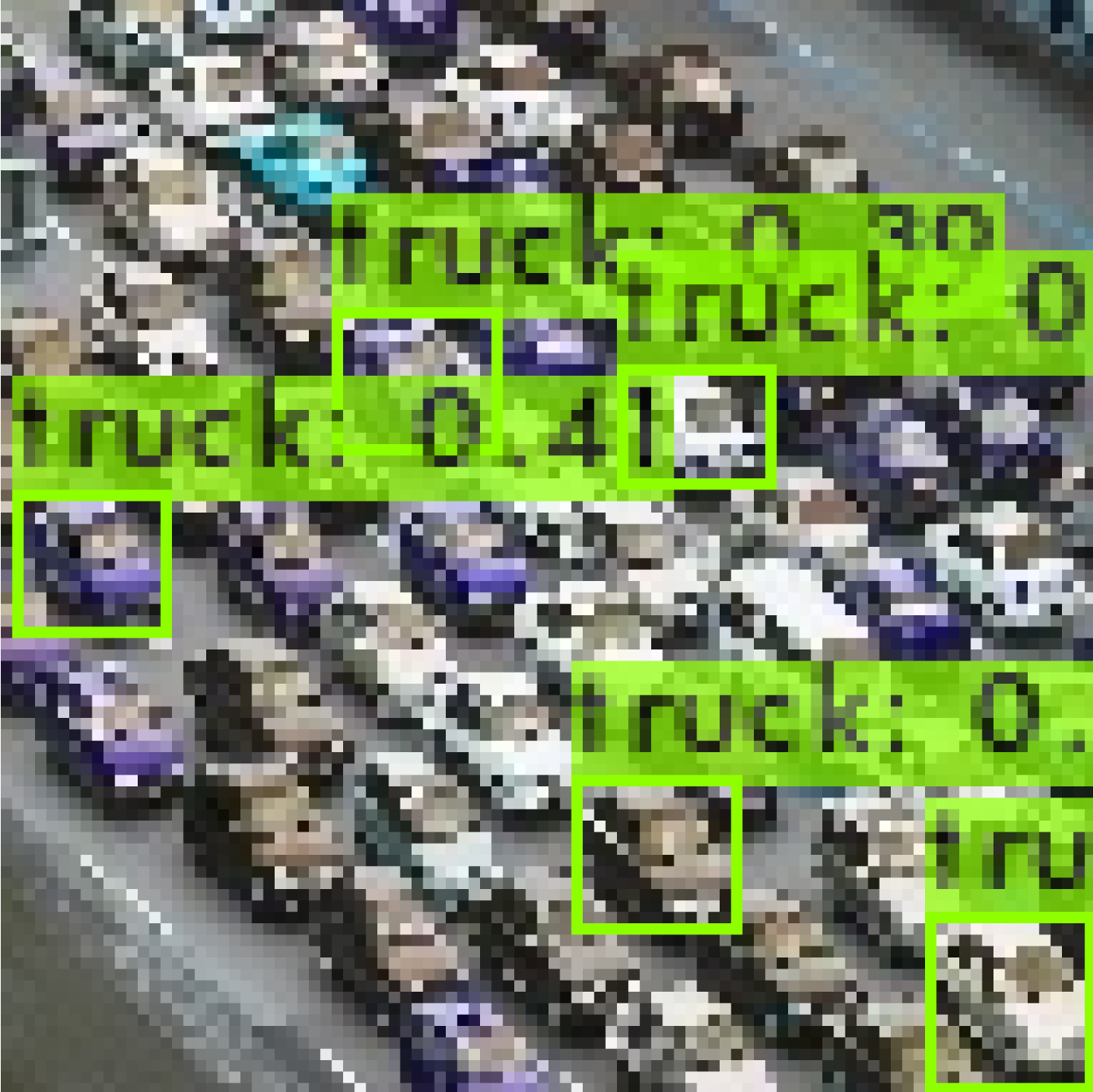
Fig 6.1 High-Resolution Image Fig 6.2 Low-Resolution Image



Fig 6.3: GENERATED IMAGE

**7.1 CHECKING CONFIDENCE WITH YOLO v4 :**

**7.1.1 Low-Resolution Image:**

****

**7.1.2 Generated Image:**

****

# 8 Conclusion:

# SRGAN has shown significant success in generating high-quality super-resolved images with finer details and improved visual fidelity compared to traditional interpolation-based methods. It has been applied in various areas, including image upscaling, video enhancement, medical imaging, and more, where high-resolution images are crucial for accurate analysis and interpretation.

# 

**9 References:**

1. Hands-on Neural Networks - Leonardo De Marchi and Laura Mitchell
2. Lawrence, J., 1993. *Introduction to neural networks*. California Scientific Software.
3. Müller, B., Reinhardt, J. and Strickland, M.T., 1995. *Neural networks: an introduction*. Springer Science & Business Media.
4. Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi."Understanding of a convolutional neural network." *2017 international conference on engineering and technology (ICET)*. Ieee, 2017.
5. Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J. and Chen, T., 2018. Recent advances in convolutional neural networks. *Pattern recognition*, *77*, pp.354-377.
6. Van Ouwerkerk, J. D. "Image super-resolution survey." *Image and vision Computing* 24.10 (2006): 1039-1052.
7. Dong, Chao, et al. "Image super-resolution using deep convolutional networks." *IEEE transactions on pattern analysis and machine intelligence* 38.2 (2015): 295-307.
8. Wang, Zhihao, Jian Chen, and Steven CH Hoi. "Deep learning for image super-resolution: A survey." *IEEE transactions on pattern analysis and machine intelligence* 43.10 (2020): 3365-3387.
9. Dong, Chao, et al. "Learning a deep convolutional network for image super-resolution." *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part IV 13*. Springer International Publishing, 2014.
10. Guo, Hui, et al. "Robust attentive deep neural network for detecting GAN-generated faces." *IEEE Access* 10 (2022): 32574-32583.
11. Zhou, F., Yang, W. and Liao, Q., 2012. Interpolation-based image super-resolution using multisurface fitting. *IEEE Transactions on Image Processing*, *21*(7), pp.3312-3318.
12. Anbarjafari, G. and Demirel, H., 2010. Image super-resolution based on interpolation of wavelet domain high-frequency subbands and the spatial domain input image. *ETRI Journal*, *32*(3), pp.390-394.
13. Li, K., Yang, S., Dong, R., Wang, X. and Huang, J., 2020. Survey of single image super‐resolution reconstruction. *IET Image Processing*, *14*(11), pp.2273-2290.
14. <https://www.tensorflow.org/api_docs/python/tf>
15. <https://www.kaggle.com/datasets/saputrahas/dataset-image-super-resolution>

￼