A MINOR PROJECT REPORT

ON

PIXART

MINOR PROJECT 2 (15B19CI591)

Submitted by:

Abhinav Verma (9917103251) F8

Lalit Garg (9917103126) F4

Neha Agrawal (9917103254) F8

Submiited to:

Dr. Raju Pal

Mrs. Varsha Garg

Mrs. Anuradha Gupta

Department of Computer Science Engineerng

Jaypee Institute of Information Technology, Noida-128,

March 2020

ACKNOWLEDGEMENT

We would like to express our special gratitude to our esteemed college, Jaypee Institute of Information Technology 128, for giving us this wonderful opportunity of making the minor project-2 of 2020, under the valuable supervision, timely suggestions and inspiration offered by our most respectable project mentor Dr. Raju Pal. It was because of his continuous encouragement that this report reached its successful completion. We also place on record and warmly acknowledge the contribution of our panel coordinators. Last but not the least we express our sincere thanks to all our friends who have patiently extended their support for accomplishing this undertaking.

Name: Neha Agarwal

Enrollment No. : 9917103254

Name: Lalit Garg

Enrollment No. : 9917103126

Name: Abhinav Verma

Enrollment No. : 9917103251

ABSTRACT

The projects title is "PixArt" in which we are trying to produce a solution for super resolution better than traditional algorithm which lacks in filling fine details and cannot remove defects and compression artifacts. For humans who carry out these tasks manually it is a very slow and painstaking process. The benefits are gaining a higher quality image from one where that never existed or has been lost, this could be beneficial in many areas or even life saving on medical applications. Also image repair and inpainting can be achieved i.e (removing of unwanted water marks from the image), repairing the defects from any type of image (jpeg compression, tears, folds and other damage). Super resolution and inpainting seem to be often regarded as separate and different tasks but by this model we can accomplish it altogether.

Contents

List of Figures

Chapter 1

INTRODUCTION

Super resolution is the process of upscaling and or improving the details within an image. Often a low resolution image is taken as an input and the same image is upscaled to a higher resolution, which is the output. The details in the high resolution output are filled in where the details are essentially unknown. Super resolution is actually is what we see in films and cinematography where we zoom in the image and image quality gets better and fine details just appears.

Through our project "PixArt" we are trying to create a solution of the above explained super resolution as traditional algorithms does not perform good and lacks in filling fine details and cannot remove defects and compression artifacts. For humans who carry out these tasks manually it is a very slow and painstaking process. So this could be beneficial in many areas or even life saving on medical applications. Also image repair and inpainting can be achieved i.e (removing of unwanted water marks from the image), repairing the defects from any type of image (jpeg compression, tears, folds and other damage). Super resolution and inpainting seem to be often regarded as separate and different tasks but by this model we can accomplish it altogether.

Figure 1.1: From left to right how quality increases.

Be it Online shopping sites, Social Media, official websites or even a childs story book, crisp and clear images can narrate the whole story of the website services or the product to the client they want to connect with their business.Since Price is highly proportional to the quality so Low Resolution Image can be a great barrier as it leads to loss in sharpness and detailing of the Image and gives a pixelated look.So it basically needs bank balance to purchase high quality cameras and mobile phones and even if you can afford it what if photography isnt your thing? Because the technique to hold and focus the camera too plays an important role in taking official photographs.Here this model is an inexpensive way to create fine detailed image from any type of low-resolution image.

Chapter 2

Background Study

Over the last two decades there has been extensive work dedicated to super resolution but all the traditional algorithm were not able to produce result of desired quality and are unable to fill details to different types of images if models works good on human images than it might or might not work good on animals and objects, also super resolution and image defect repairing, compression artifacts cannot be performed together by same model. So, in order to generate suitable results we have created U-Net architecture with cross connection similar to denseNet.

2.1 U-Net architecture with cross connections similar to a DenseNet

A U-Net[3] is a convolutional neural network architecture that was developed for biomedical image segmentation. U-Nets have been found to be very effective for tasks where the output is of similar size as the input and the output needs that amount of spatial resolution. This makes them very good for creating segmentation masks and for image processing/generation such as super resolution.When convolutional neural nets are commonly used with images for classification, the image is taken and downsampled into one or more classifications using a series of stride two convolutions reducing the grid size each time.To be able to output a generated image of the same size as the input, or larger, there needs to be an upsampling path to increase the grid size. This makes the network layout resemble a U shape, a U-Net the downsam-

pling/encoder path forms the left hand side of the U and the upsampling/decoder path forms the right hand part of the U-net.

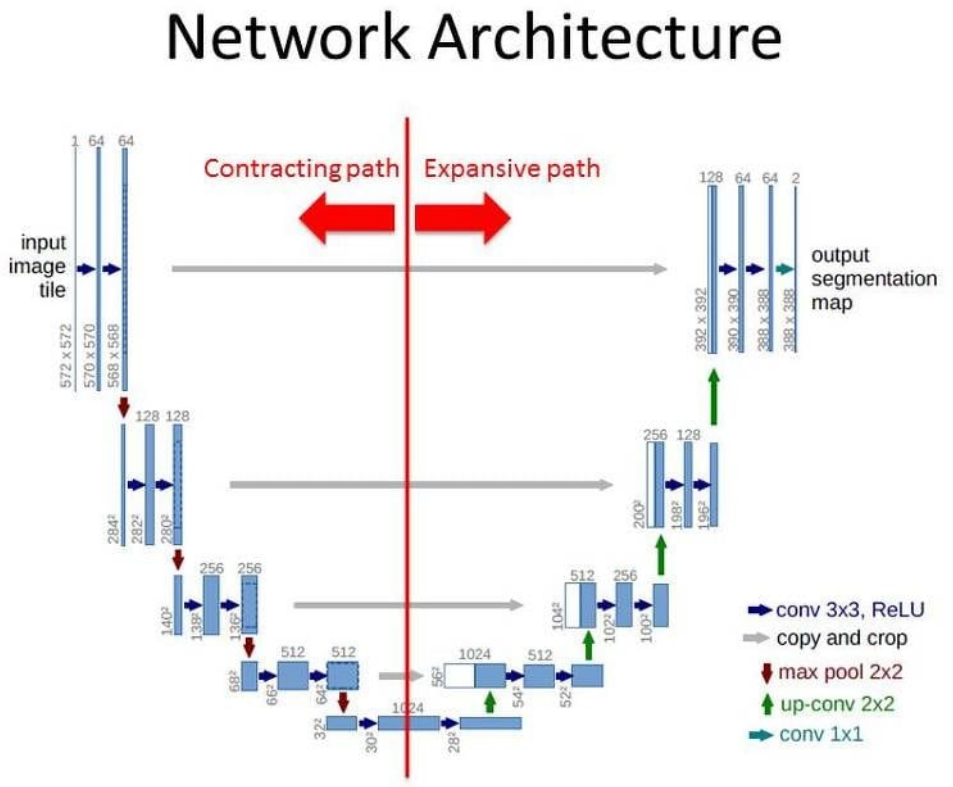


Figure 2.1: U-Net architecture of the model.

2.2 Residual blocks (ResBlocks) and dense blocks

Convolutional networks[1] can be substantially deeper, more accurate, and more efficient to train if they contain shorter connections between layers close to the input and those close to the output. If you visualise the loss surface (the search space for the varying loss of the models prediction), this looks like a series of hills and valleys as the left hand image in the diagram below shows. The lowest loss is the lowest point. Research has shown that a smaller optimal network can be ignored even if its an exact part of a bigger network. This is because the loss surface is too hard to navigate. This means that by adding layers to the model it can make the prediction become worse. A solution thats been very effective is to add cross connections between layers of the network allowing large sections to be skipped if needed. This creates a loss surface

that looks like the image on the right. This is much easier for the model to be trained with optimal weights to reduce the loss.

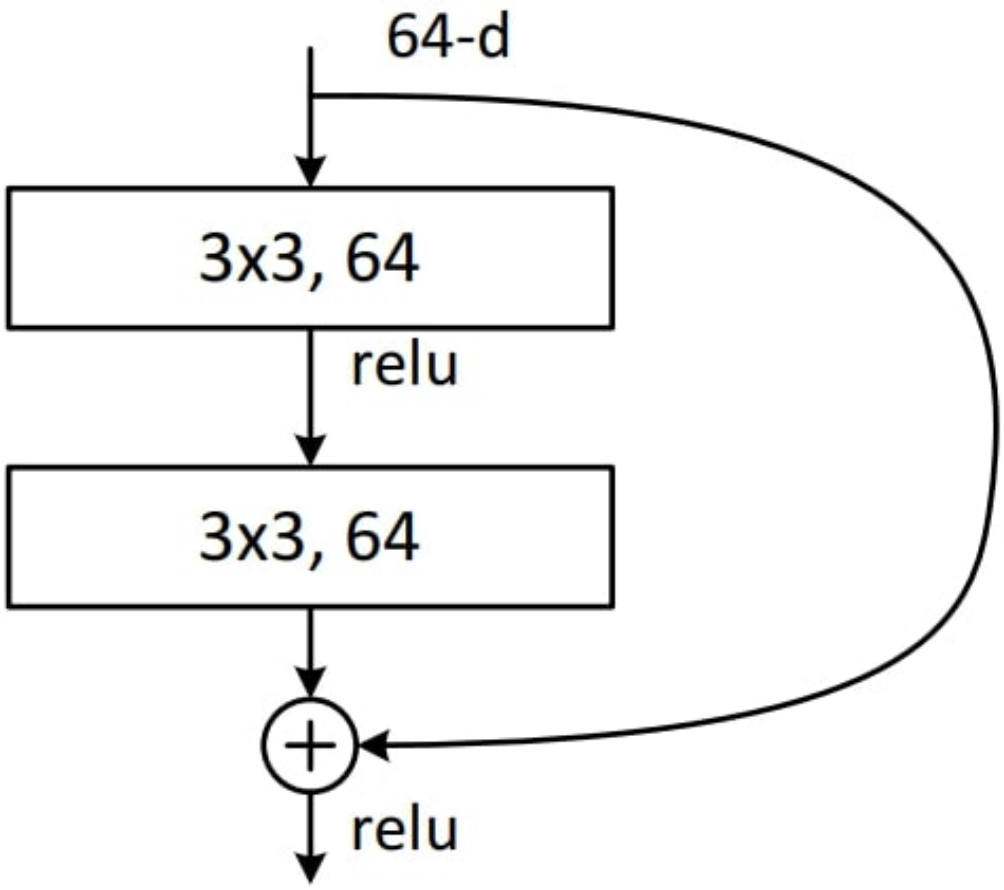


Figure 2.2: ResNet-34 with Skip connection.

2.3 VGG-16 Network Architecture

VGG is [2]another CNN architecture devised in 2014, the 16 layer version is utilised in the loss function for training this model.The VGG model. a network pretrained on ImageNet, is used to evaluate the generator models loss. Normally this would be used as a classifier to tell you what the image is, for example is this a person, a dog or cat. The head of the VGG model is ignored and the loss function uses the intermediate activations in the backbone of the network, which represent the feature detections. The head and backbone of networks are described a little further in the training section further on.

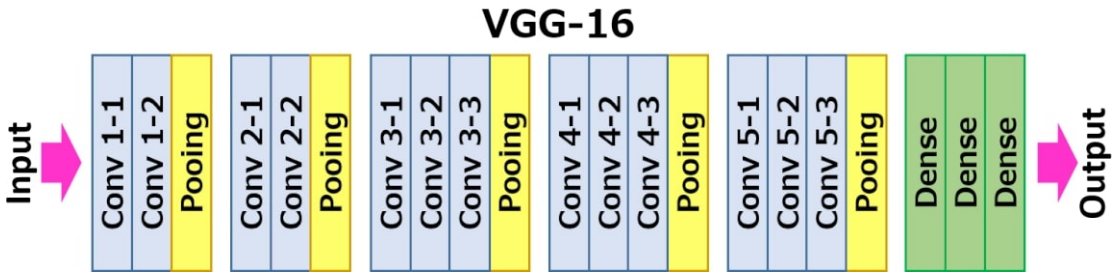


Figure 2.3: Loos Function using VGG-16

2.4 GAN Network Architecture

SRGAN uses the GAN [4]to produce the high resolution images from the low resolution images. In this implementation, a 64 X 64 image is converted into the 256 X 256 image using the concept of GAN. For training a high resolution image is downsampled into low resolution image using Gaussian blur followed by resizing it to 64 X 64. As a whole, SRGAN upsample the images by a factor of 4 producing high resolution images. A Generator is used to generate 256 X 256 images from 64 X 64 images and a discriminator.

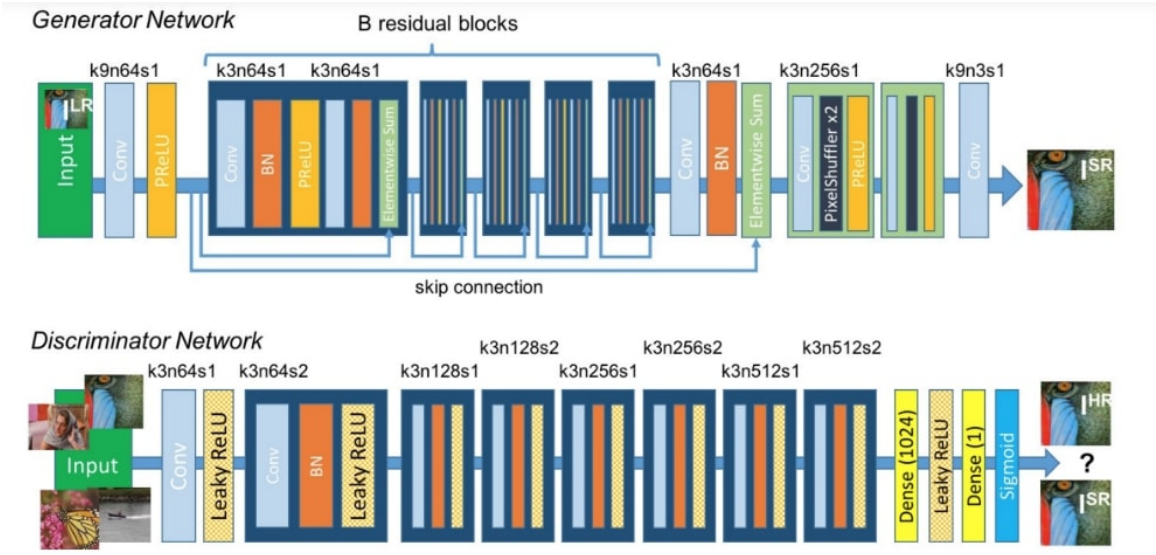


Figure 2.4: GAN architecture to perform super resolution.

Chapter 3

Requirement Analysis

The project aims to implement high level machine learning algorithm which require different software and hardware which is mentioned below.

1. **Hardware:** The model will require 12GB Gpu and 16GB ram for smooth training and working.

2. **Software Requirements:** The model require many different software library like tensorflow, keras, Numpy, Pytorch, ResNet34, VGG16, SRGAN, and python 3.0 or above.

3. **Dataset Requirements:** Model is build using Div2K super resolution datasets, cifar10, and ImageNet.

Chapter 4

PixArt

The projects title is "PixArt" in which we are trying to produce a solution for super resolution better than traditional algorithm which lacks in filling fine details and cannot remove defects and compression artifacts. For humans who carry out these tasks manually it is a very slow and painstaking process. The benefits are gaining a higher quality image from one where that never existed or has been lost, this could be beneficial in many areas or even life saving on medical applications. Also image repair and inpainting can be achieved i.e (removing of unwanted water marks from the image), repairing the defects from any type of image (jpeg compression, tears, folds and other damage). Super resolution and inpainting seem to be often regarded as separate and different tasks but by this model we can accomplish it altogether.

4.1 Detailed Design

A U-Net is a convolutional neural network architecture that was developed for biomedical image segmentation. U-Nets have been found to be very effective for tasks where the output is of similar size as the input and the output needs that amount of spatial resolution. This makes them very good for creating segmentation masks and for image processing/generation such as super resolution. When convolutional neural nets are commonly used with images for classification, the image is taken and downsampled into one or more classifications using a series of stride two convolutions reducing the grid size each time.

To be able to output a generated image of the same size as the input, or larger, there needs to be an upsampling path to increase the grid size. This makes the network layout resemble a U shape, a U-Net the downsampling/encoder path forms the left hand side of the U and the upsampling/decoder path forms the right hand part of the Unet. For the upsampling/decoder path several transposed convolutions accomplishes this, each adding pixels between and around the existing pixels. Essentially the reverse of the downsampling path is carried out. The options for the upsampling algorithms are discussed further on. This model or mathematical function has over 40 million parameters or coefficients allowing it to attempt to preform super-resolution.

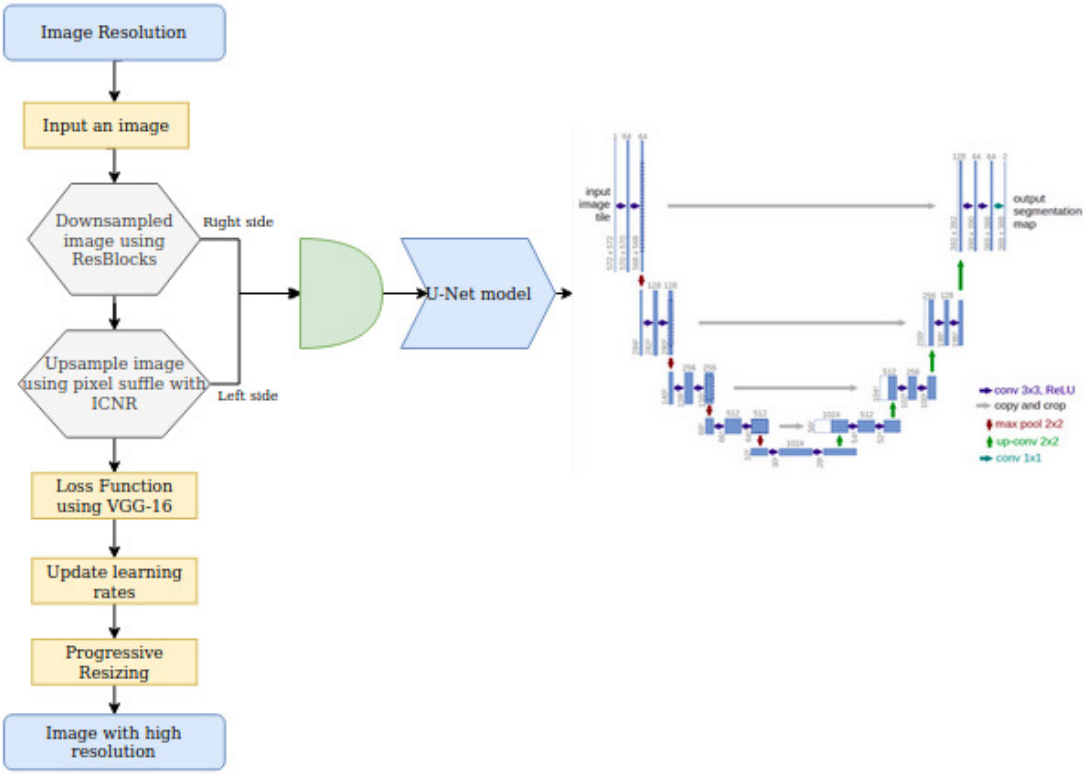


Figure 4.1: Flowchart of complete model.

Chapter 5

IMPLEMENTATION

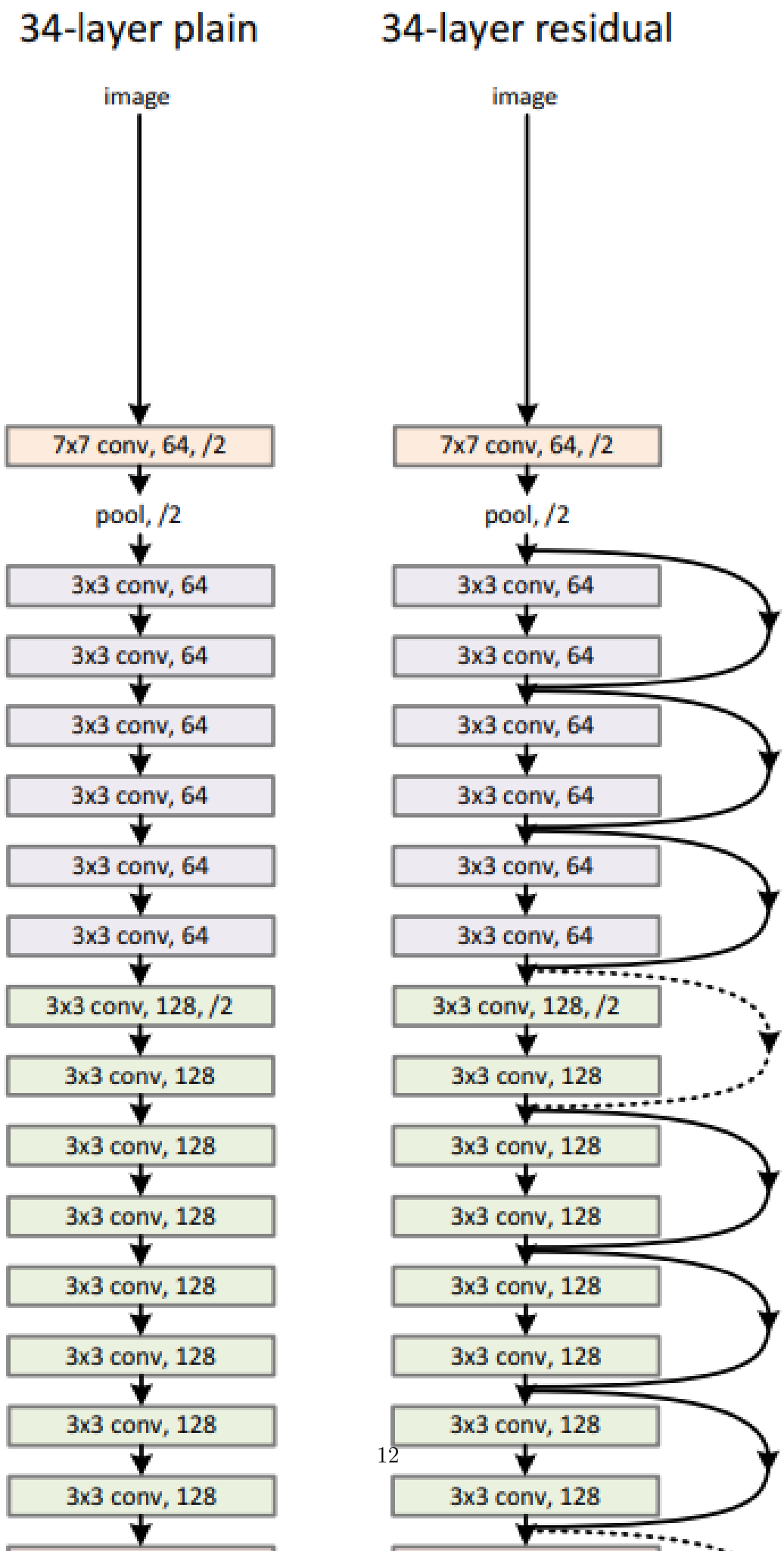
The low-resolution image is initially a copy of the target/ground truth image at half the dimensions and then further low-resolution image data is augmented to train the model to learn each and every possible deformity.

The method uses the following mathematical function models:

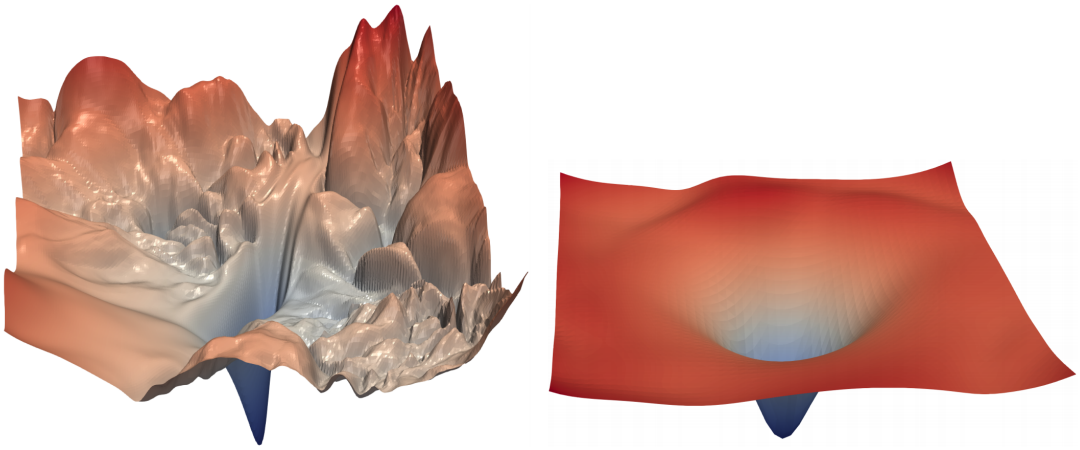
1. A U-Net architecture with cross-connections similar to a DenseNet.
2. A ResNet-34 based encoder and a decoder based on ResNet-34.
3. Pixel Shuffle upscaling with ICNR initialization.
4. Transfer learning from pre-trained ImageNet models.
5. A loss function based on activations from a VGG-16 model, pixel loss and gram matrix loss.
6. Discriminative learning rates.
7. Progressive resizing.

5.1 ResNet-34

ResNet is a Convolutional Neural Network (CNN) architecture, made up of series of residual blocks (ResBlocks) described below with skip connections differentiating ResNets from other CNNs. When first devised ResNet won that years ImageNet competition by a significant margin as it addressed the vanishing gradient problem, whereas more layers are added training slows and accuracy doesnt improve or even gets worse. It is the networks skip connections that accomplish this feat.


34-layer plain
34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128

Convolutional networks can be substantially deeper, more accurate, and more efficient to train if they contain shorter connections between layers close to the input and those close to the output. If you visualize the loss surface (the search space for the varying loss of the models prediction), this looks like a series of hills and valleys as the left-hand image in the diagram below shows. The lowest loss is the lowest point. Research has shown that a smaller optimal network can be ignored even if its an exact part of a bigger network. This is because the loss surface is too hard to navigate. This means that by adding layers to the model it can make the prediction become worse.



(a) without skip connections (b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

Figure 5.2: loss before and after skip connection.

5.2 U-Net Architecture

A [3]U-Net is a convolutional neural network architecture that was developed for biomedical image segmentation. U-Nets have been found to be very effective for tasks where the output is of similar size as the input and the output needs that amount of spatial resolution. This makes them very good for creating segmentation masks and for image processing/generation such as super-resolution To be able to output a generated image of the same size as the input, or larger, there needs to be an upsampling path to increase the grid size. This makes the network layout resemble a U shape, a U-Net the downsampling/encoder path forms the left-hand side of the U

and the upsampling/decoder path forms the right-hand part of the U.

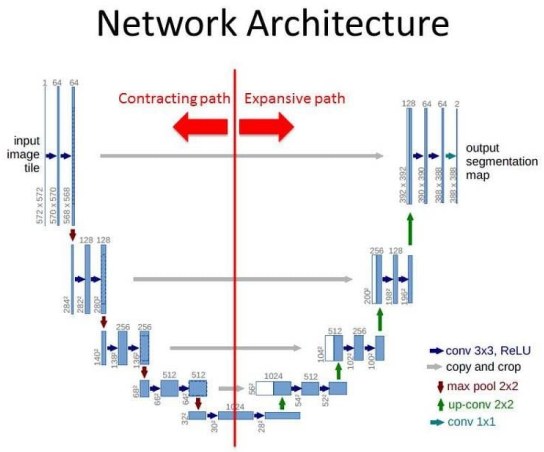


Figure 5.3: U-Net Architecture

5.3 VGG Loss funtion

This model used here is trained with a similar loss function to the paper, using VGG16 but also combined with pixel mean squared error loss and gram matrix loss. VGG16 is another CNN architecture devised in 2014, the 16 layer version is utilized in the loss function for training this model.

The VGG model. a network pretrained on ImageNet, is used to evaluate the generator models loss. Normally this would be used as a classifier to tell you what the image is, for example is this a person, a dog or cat. The training of this super-resolution model uses the loss function based on the VGG models activations. The loss function remains fixed throughout the training unlike the critic part of a GAN. The Feature map has 256 channels by 28 by 28 which are used to detect features such as fur, eyeball, wings and the type material among many other type of features. The activations at the same layer for the (target) original image and the generated image are compared using mean squared error or the least absolute error (L1) error for the base loss. These are feature losses. This error function uses an L1 error. This allows the loss function to know what features are in the target ground truth image and to evaluate how well the models predictions features match these rather than only comparing pixel difference.

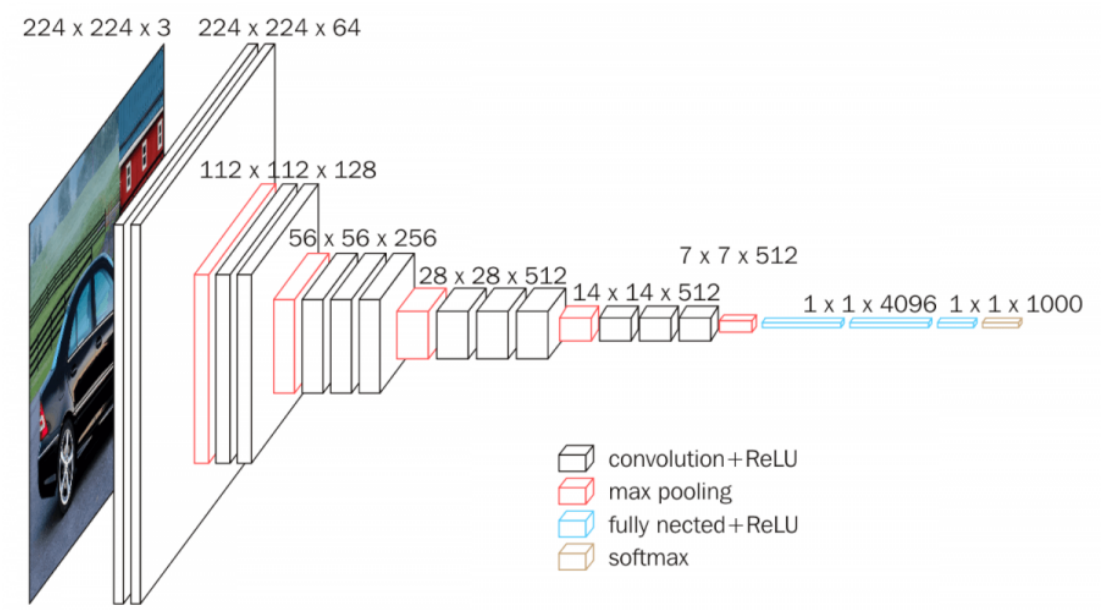


Figure 5.4: Loss function using VGG16

5.4 Training

The training process begins with a model as described above: a U-Net based on the ResNet-34 architecture pre-trained on ImageNet using a loss function based on the VGG-16 architecture pre-trained on ImageNet combined with pixel loss and a gram matrix. Training the head and the backbone of the model

```
atagen = ImageDataGenerator(
        featurewise_center=False,
        samplewise_center=False,
        featurewise_std_normalization=False,
        samplewise_std_normalization=False,
        zca_whitening=False,
        zca_epsilon=1e-06,
        rotation_range=0,
        width_shift_range=0.1,
        height_shift_range=0.1,
        shear_range=0.,
        zoom_range=0.,
        channel_shift_range=0.,
        fill_mode='nearest',
        cval=0.,
        horizontal_flip=True,
        vertical_flip=False,
        rescale=None,
        preprocessing_function=None,
        data_format=None,
        validation_split=0.0)

atagen.fit(X_train)
odel1.fit_generator(datagen.flow(X_train, Y_train, batch_size=BATCH_SIZE),validation_data=(X_test, Y_test),epochs=EPOCHS,
```

Figure 5.5: Data Augmentation

Three methods used here in particular help the training process. These are pro-

gressive resizing, freezing then unfreezing the gradient descent update of the weights in the the backbone and discriminative learning rates. The models architecture is split into two parts, the backbone and the head. The backbone is the left hand section of the U-Net, the encoder/down-sampling part of the network based on ResNet-34. The head is the right hand section of the U-Net, the decoder/up-sampling part of the network. The backbone has pretrained weights based on ResNet34 trained on ImageNet, this is the transfer learning. The head needs its weights training as these layers weights are randomly initialised to produce the desired end output. At the very start the output from the network is essentially random changes of pixels other than the Pixel Shuffle sub-convolutions with ICNR initialisation used as the first step in each upscale in the decoder/upsampling path of the network. Once trained the head on top of the backbone allows the model to learn to do something different with its pretrained knowledge in the backbone. Freeze the backbone, train the head The weights in the backbone of the network are frozen so that only the weights in the head are initially being trained. A learning rate finder is run for 100 iterations and plots the graph of loss against learning rate, a point around the steepest slope down towards the minimum loss is selected as the maximum learning rate. Alternatively a rate 10 times less than the lowest point can be used to see if that performs any better.

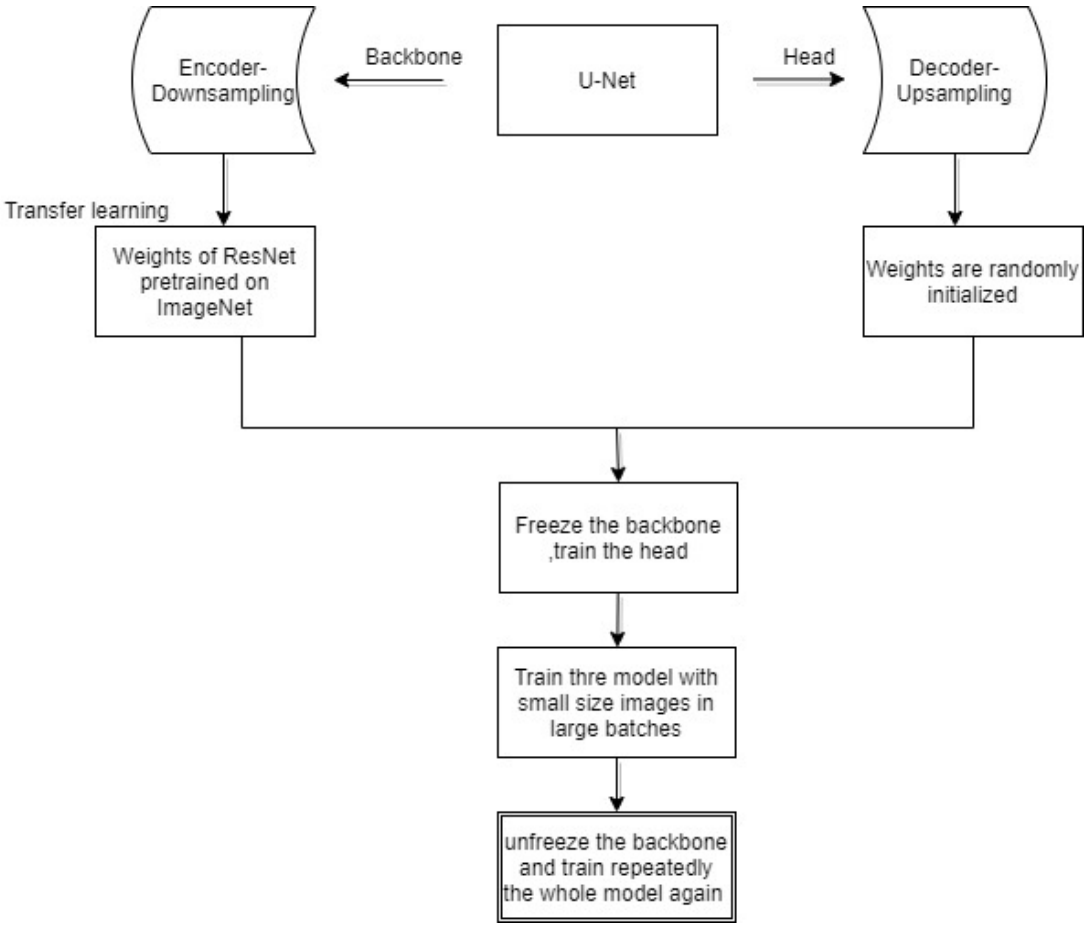


Figure 5.6: Training flowchart

Chapter 6

Results and Analysis

```
#ResNet34

in_layer = Input(shape=[32, 32, 3])
conv_1=Conv2D(64, kernel_size=(7, 7), strides=(2, 2), padding='same')(in_layer)
pool1=MaxPool2D(pool_size=(3, 3), strides=(2, 2), padding='same')(conv_1)
x=Conv2D(64, kernel_size=(3, 3), strides=(2, 2), padding='same')(pool1)
x =Conv2DTranspose(64,(4,4),padding='same',activation='relu',strides=2)(x)

for i in range(5):
  x=Conv2D(64, kernel_size=(3, 3), padding='same')(x)

for i in range(8):
  x=Conv2D(128, kernel_size=(3, 3), strides=(2, 2), padding='same')(x)

for i in range(12):
  x=Conv2D(256, kernel_size=(3, 3), strides=(2, 2), padding='same')(x)

for i in range(6):
  x=Conv2D(512, kernel_size=(3, 3), strides=(2, 2), padding='same')(x)

x=GlobalAveragePooling2D()(x)
x=Dense(10,activation='softmax',kernel_initializer='he_normal')(x)
model = Model(inputs=[in_layer], outputs=[x])
model.compile(loss='categorical_crossentropy',optimizer=Adam(lr=lr_schedule(0)),metrics=['accuracy'])

model.summary()
model1=model
```

Figure 6.1: ResNet34 code screenshot

```
25000/25000 [==============================] - 19s 759us/step
[0.4609957277488709, 0.79816]
```

Figure 6.2: ResNet Accurany

```
#ResBlock34

in_layer = Input(shape=[32, 32, 3])

d1=Conv2D(64, kernel_size=(7, 7) , padding='same')(in_layer)
pool1=MaxPool2D(pool_size=(3, 3) , padding='same')(d1)
d2=Conv2D(64, kernel_size=(3, 3) , padding='same')(pool1)
d3=Conv2D(64, kernel_size=(3, 3) , padding='same')(d2)
#Skip connection
s1 = Add()([pool1,d3])
d4=Conv2D(64, kernel_size=(3, 3) , padding='same')(s1)
d5=Conv2D(64, kernel_size=(3, 3) , padding='same')(d4)
#Skip connection
s2 = Add()([d5,s1])
d6=Conv2D(64, kernel_size=(3, 3) , padding='same')(s2)
d7=Conv2D(64, kernel_size=(3, 3) , padding='same')(d6)
#Skip connection
s3 = Add()([d7,s2])
d8=Conv2D(128, kernel_size=(3, 3) , padding='same')(s3)
d9=Conv2D(128, kernel_size=(3, 3) , padding='same')(d8)
s4 = d9
d10=Conv2D(128, kernel_size=(3, 3) , padding='same')(s4)
d11=Conv2D(128, kernel_size=(3, 3) , padding='same')(d10)
#Skip connection
s5 = Add()([d11,s4])
d12=Conv2D(128, kernel_size=(3, 3) , padding='same')(s5)
d13=Conv2D(128, kernel_size=(3, 3) , padding='same')(d12)
#Skip connection
s6 = Add()([d13,s5])
d14=Conv2D(128, kernel_size=(3, 3) , padding='same')(s6)
d15=Conv2D(128, kernel_size=(3, 3) , padding='same')(d14)
#Skip connection
s7 = Add()([d15,s6])
d16=Conv2D(256, kernel_size=(3, 3) , padding='same')(s7)
d17=Conv2D(256, kernel_size=(3, 3) , padding='same')(d16)
s8 =d17
d18=Conv2D(256, kernel_size=(3, 3) , padding='same')(s8)
d19=Conv2D(256, kernel_size=(3, 3) , padding='same')(d18)
#Skip connection
s9 = Add()([d19,s8])
d20=Conv2D(256, kernel_size=(3, 3) , padding='same')(s9)
d21=Conv2D(256, kernel_size=(3, 3) , padding='same')(d20)
s10 = Add()([d21,s9])
d22=Conv2D(256, kernel_size=(3, 3) , padding='same')(s10)
d23=Conv2D(256, kernel_size=(3, 3) , padding='same')(d22)
```

Figure 6.3: ResBlock code screenshot

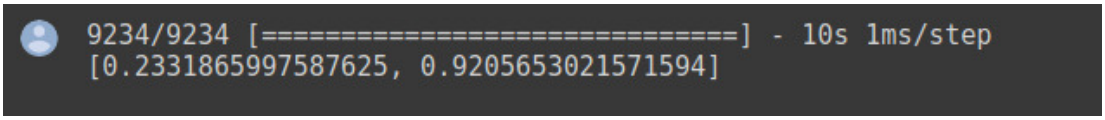


Figure 6.4: ResBlock Accuracy

Chapter 7

Conclusion and Future Scope

Till now we have went through different research papers and implemented ResNet and ResBlock for backbone of the U-net architecture and concluded that resNet with skip connections(ResBlock) performed better than normal ResNet34 having accuracy 0.92 and 0.79 respectively and now for further implementation we will use ResBlock and perform the scopes given below.

The major objective of the project is to upscale and improve the quality of the image by filling the details where essentially are unknown, now can be filled using the model. U-Net deep learning based super resolution trained using loss functions such as these can perform very well for super resolution including: Furure Scope of this project

1. Upscaling low-resolution images to higher resolution images.
2. Improving the quality of an image maintaining the resolution.
3. Removing watermarks.
4. Removing damaging from images.
5. Removing JPEG and other compression artifacts.
6. Colorizing greyscale images.

Chapter 8

Limitations

1. If the model is trained on Animals, then its not likely the model will perform well on a completely different dataset category such as room interiors or flowers.

2. The results of super resolution on models trained on close up human faces werent particularly convincing although.

3. For very low resolution images or those with a lot of compression artifacts, this may still be preferable. This is an area can plan to continue to explore.

4. Graying of color image and vice-versa is still can't be achieved. 5. Also the hardware required for the purpose is expensive.

Bibliography

[1] Michal Drozdzal, Eugene Vorontsov, Gabriel Chartrand, Samuel Kadoury, and Chris Pal. The importance of skip connections in biomedical image segmentation. In *Deep Learning and Data Labeling for Medical Applications*, pages 179–187. Springer, 2016.

[2] Hussam Qassim, Abhishek Verma, and David Feinzimer. Compressed residual-vgg16 cnn model for big data places image recognition. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 169–175. IEEE, 2018.

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[4] Hoo-Chang Shin, Neil A Tenenholtz, Jameson K Rogers, Christopher G Schwarz, Matthew L Senjem, Jeffrey L Gunter, Katherine P Andriole, and Mark Michalski. Medical image synthesis for data augmentation and anonymization using generative adversarial networks. In *International workshop on simulation and synthesis in medical imaging*, pages 1–11. Springer, 2018.