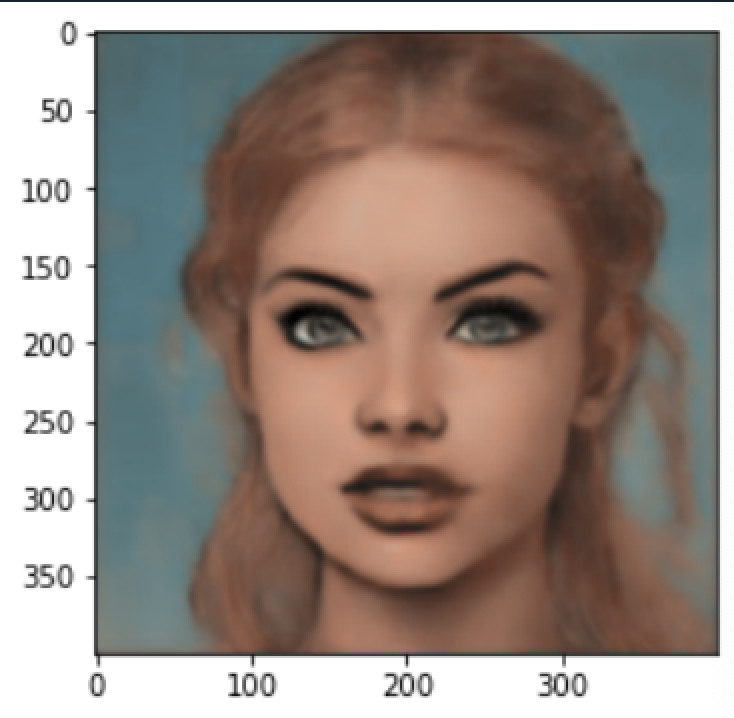
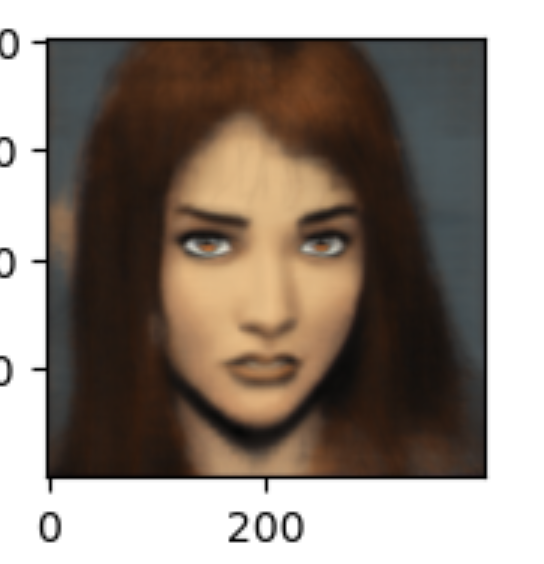
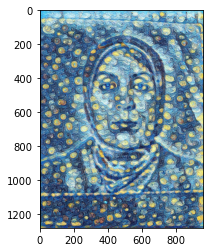
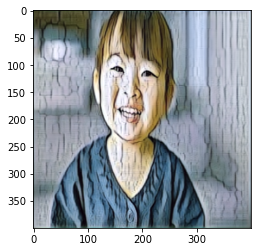


ART BY AI

**Project Doucumintation**

**Prepared by :**

**Jwaher Alwahidi**

**By Python Class**

Table of Contents

1- Introduction ........................................................................................................................ 5

1.1 Background ................................................................................................................... 5

1.2 Aim and objectives ....................................................................................................... 5

2- Methodology....................................................................................................................... 6

2.1 What is CNN model ?...................................................................................................... 6

2.2 What is U-net CNN Model?............................................................................................. 8

2.3 U-Net Model Layers ........................................................................................................ 9

2.4 U-Net Model Code………………................................................................................. 10

3- Data Proccising................................................................................................................... 19

4- Result..................................................................................................................................

4.1Models outputs................................................................................................................ 20

4.1Output loss diagram......................................................................................................... 21

5-Data Sources ......................................................................................................................... 22

6-References............................................................................................................................. 23

**List of Figures**

Figure 1: CNN classification model ……………………………….............................................. 7

Figure 2: Calculating wight and error using back prpagation ………………................................. 7

Figure 3: image segmentation........................................................................................................... 8

Figure 4: U-shape for U-net.............................................................................................................. 9

Figure 5: Convoultion kernal (3,3).................................................................................................. 10

Figure 6: Activation functions.......................................................................................................... 11

Figure 7: Zero padding...................................................................................................................... 11

Figure 8: Causal padding.................................................................................................................. 12

Figure 9: Valid padding...................................................................................................................... 12

Figure 10: Dropout............................................................................................................................ 13

Figure 11: pooling method effects ………………………………………………………………….13

Figure 12: Conv2DTranspose Vs Conv2D........................................................................................ 15

Figure 13: Concatenate layers ........................................................................................................... 16

Figure 14: loss diagram....................................................................................................................... 21

**ABSTRACT**

In the past sketching or creating a cartoon need to be done by hand. Nowadays, many applications can turn your photos to cartoons or to art. this kind of photo is common recently on social networking sites and around the world. But what if I told you, that you can create your own effect with just a few lines of code? Using the Convolutional Neural Network and the U-Net model, in this project, we will be able to produce 4 types images. By using U-net, three of these photos will be generate and the fourth one will be generated by using Tensor Flow Hub.

**INTRODUCTION**

* 1. Background

As we know that a computer cannot understand words or pictures, it only understands numbers. Therefore, we must first convert the images into numbers, and then enter these numbers on the Neural Network, in order to carry out some operations, and then produce other new images.

These days, there are many applications, such as LENSA, which converts images with high quality, and it has gained wide popularity recently because it makes features more beautiful and eliminates some kind of modifications to features to become more beautiful.

1.2 Aim and objectives

The goal of the project is to learn how to deal with images and learn the stages of image processing as inputs and outputs, through:

1) The first model is a model that combines two images, as the first image is a personal image known as “content image”, while the second image is a picture of a painting known as “Style image” , then the two images will be merged to produce one image

2) The second model is a model that can transform a normal image to look like a painting, by using U-net model, which we will talk about later in this document.

3) The third model is a model that can transform the image to look like a cartoon character using U-net as well

4) The fourth model is a model that combines cartoon and reality, so that the image appears to be a mixture between the two as well, by using U-net model

**Methodology**

**2.1 What is CNN model ?**

Vision is one of the most important senses that humans possess, and the ability to capture reflected light rays and extract meaning from them is a very complex task. Can we give machines the same ability in a very short period of time? For computers, as we mentioned, these images are nothing but matrices containing many numbers, and understanding the subtle differences between matrices has been a complex issue for many mathematicians for years. But after the advent of artificial intelligence, especially CNN engineering, research has made unprecedented progress. Many problems that were previously considered untouchable are now showing amazing results.

CNN(Convolutional neural networks ) is a kind of deep learning model for image processing,it is a mathematical construct that usually consists of three types of layers convolution, pooling, and fully connected layers. The first two layers, convolution and pooling layers, extract features, and the third layer maps the extracted features into the final output, such as classification.

The convolution layer(Conv2D) plays a major role in CNN, which consists of a stack of arithmetic operations, such as convolution, which is a specialized type of linear operation. As shown in figure (1) In digital images, pixel values are stored in a two-dimensional (2D) grid as set of numbers, and a small grid of parameters called the kernel is applied, which is an optimizable feature. at each image position, which makes CNNs highly efficient for image processing, as a feature may appear anywhere in the image.

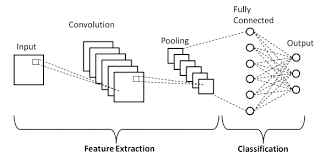


Figure (1)

When one layer feeds its output into the next layer, the extracted features can become hierarchically and progressively more complex. The process of optimizing parameters such as the kernel is called training, which is performed to reduce the divergence between the output and ground truth labels through an optimization algorithm called backpropagation and gradient de-scent, among others as shown in figure (2)

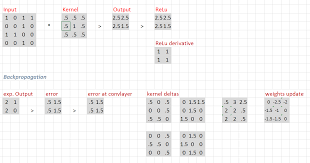


Figure (2)

**2.2 What is U-net CNN Model ?**

One of the problems with image proccicing was image segmentation. In image segmentation, the device has to divide the image into parts and each part has different qualities

As the figure (3) shows, the image has been transformed into two parts, the cat and the background.



Figure (3)

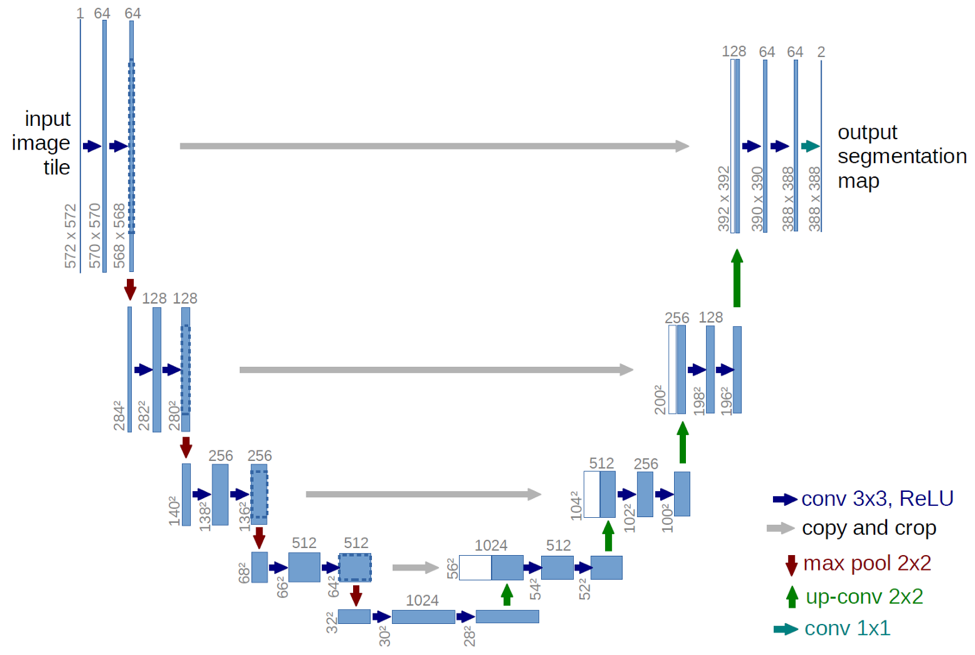
Convolutional neural networks gave good results on simpler image segmentation problems but did not make any good progress on complex problems. This is where UNet comes into the picture. UNet was first designed specifically for medical image segmentation. It showed good results, which I then used in many other fields.

**The intuition behind Unet :**

The main idea behind CNN is to learn image feature mapping and use it to make more accurate feature maps. This works well for classification problems as the image is converted to class ouput. But in image segmentation, we need to not only convert the feature map into a vector but also regenerate an image from this vector so the output is another image. This is a huge task because it is much more difficult to convert a vector into an image than just classify the image. The whole idea of UNet revolves around this problem.

While converting an image to a vector, byusing Conv2D and max pooling to feature map the image, so we use the same feature maps used for shrinkage to expand a vector into a segmented image. This would keep the same structural parts of the image which mean less distortion.

## **2.3 U-Net Model Layers**

  
Figure (4): U-shape for U-net

U-Net Architecture:

The architecture looks like the letter "U" as shown in figure(4) , that’s why we called it U-net which is consists of three sections: contraction, throttling, and expansion section. The shrinkage section consists of many shrinkage blocks. Each block takes an input that applies two 3x3 Conv2D layer followed by a 2x2 max pooling layer. The number of nurans , or feature maps, doubles after each block so that the architecture can learn complex structures effectively. The bottom layer is intermediate between the shrinkage layer and the expansion layer.

So the steps in the U-net model are :

**2.4 U-Net Model Code**

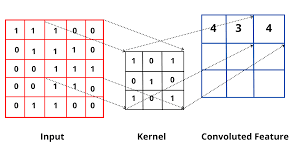
**Let’s look deeper to our model:**

**First step**: Down sampling path(Encoder Path):

|  |
| --- |
| c1 = Conv2D(16, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (inputs)  c1 = Dropout(0.1) (c1)  c1 = Conv2D(16, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (c1)  p1 = MaxPooling2D((2, 2)) (c1) |

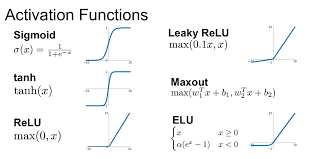
1. Conv2D:

* This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs , which (3,3) represent the size of kernel as shown in figure(5)

  
Figure (5)

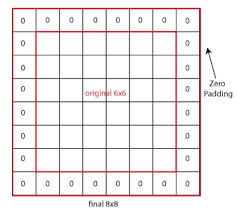
* Where the activation function depends on what is the output of the model ,each activation function have range of number as shown in figure (6) , so if your model is binary classification model so the activation function should be tanh because the output is binary (-1,1) ,etc.

In our model the numbers between (0,inf) because each pixel represented by certain number so our options are (Leaky ReLU , ReLU,ELU) but due to the limited computer power we will use ELU because it is faster than ReLU

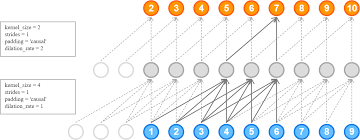
  
figure (6)

* In this model the wight initializer: ‘ he\_normal’.
* Padding: it adds some extra space to cover the image which helps the kernel to improve performance. So, this technique is dealing with the edges of the photo. there are three types of padding technique:

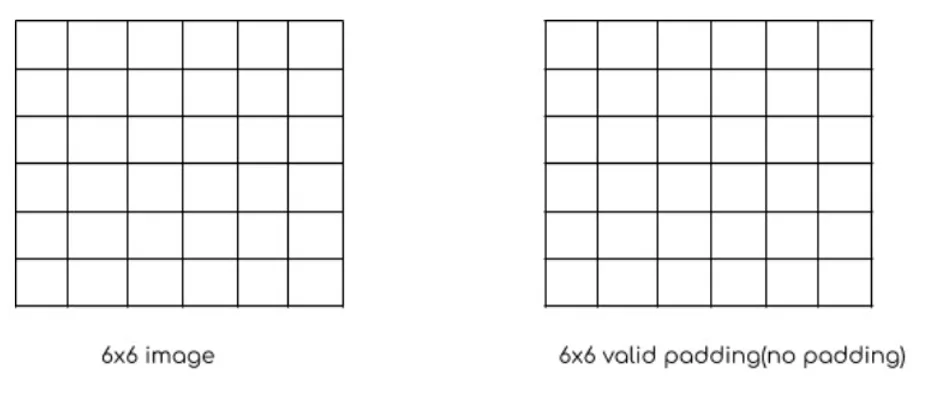
1. Same padding: it is used when we need an output of the same shape as the input. This value calculates and adds padding required to the input image to ensure the shape before and after. If the values for the padding are zeroes then it can be called zero padding. When the padding is set to zero, then every pixel in padding has value of zero as shown in figure (7)

  
Figure (7)

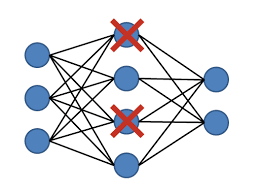
1. Causal padding: it is padding where it predicts new edges as shown in figure (8)

  
Figure (8)

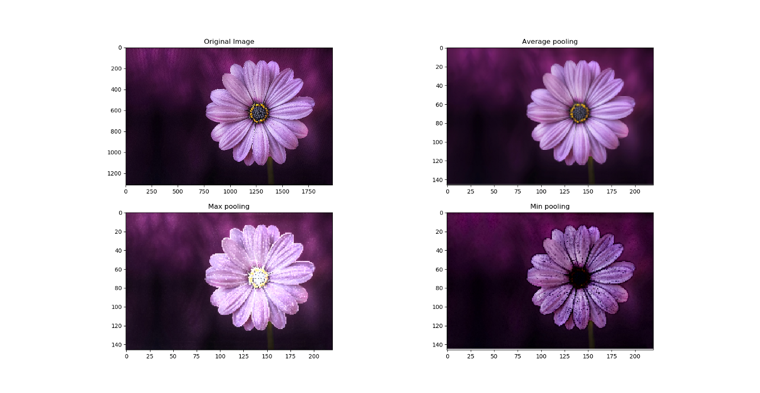
1. Valid padding: no padding by default keras choose if not specified . shown in figure (9)

  
Figure (9)

1. Dropout : The term "dropout" is used for a technique that excludes some network nodes. In which the neurons of the network whose weight is a certain number chosen by the programmer are ignored as shown in figure (10). This technique is applied in the training phase to reduce the effects of overfitting and the network can concentrate on other features

  
Figure (10)

1. Maxpooling : there is three types of pooling operations,each one effect the output photo as shown in figure (11) :
2. Max pooling: The maximum pixel value of the batch is selected.
3. Min pooling: The minimum pixel value of the batch is selected.
4. Average pooling: The average value of all the pixels in the batch is selected.

  
Figure (11)

Choosing the appropriate pooling depends on the type of data, as the average pooling method works to smooth the image and thus sharp features may not be identified when using it, the Min pooling works to take the lowest value, which reduces the clarity and quality of the image, but the Max pooling works to take the highest value, which increases the clarity and sharpness of the features .As for the reason for choosing (2,2) , we want the largest number of features to help the resulting image be as close as possible to the expected result.

The first step will repeated 3 more times as shown:

|  |
| --- |
| c2 = Conv2D(32, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (p1) c2 = Dropout(0.1) (c2) c2 = Conv2D(32, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (c2) p2 = MaxPooling2D((2, 2)) (c2)c3 = Conv2D(64, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (p2) c3 = Dropout(0.2) (c3) c3 = Conv2D(64, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (c3) p3 = MaxPooling2D((2, 2)) (c3)c4 = Conv2D(128, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (p3) c4 = Dropout(0.2) (c4) c4 = Conv2D(128, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (c4) p4 = MaxPooling2D(pool\_size=(2, 2)) (c4) |

Then 2 conv layers without max pooling as shown:

|  |
| --- |
| c5 = Conv2D(256, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (p4) c5 = Dropout(0.3) (c5) c5 = Conv2D(256, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (c5) |

Which is the end of Down sampling path process ,where the output size is 25x25x256

**second step**: Expansion/Up sampling path(Decoder Path):

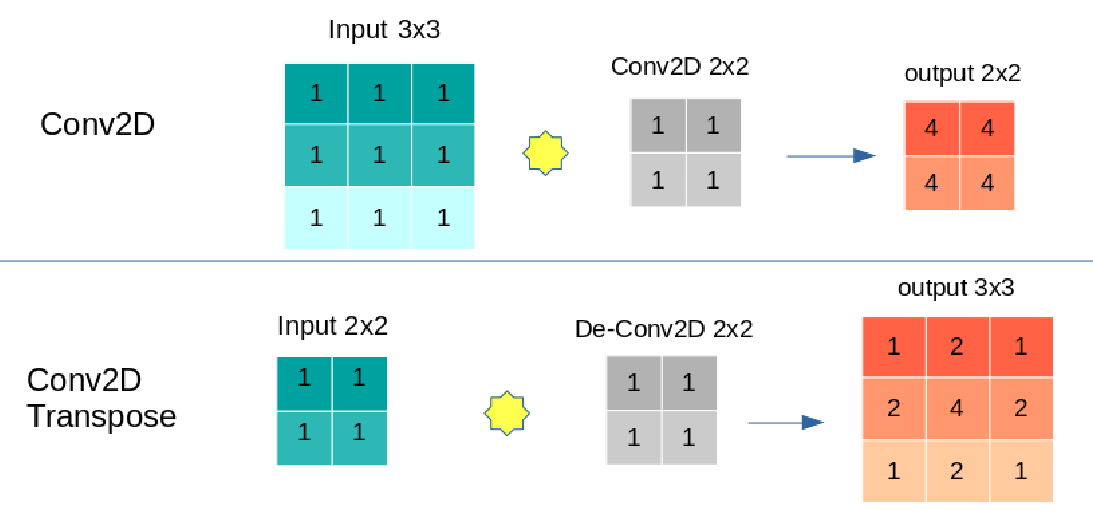
The expansion path consists of 4 blocks. Each block consists of:

Let’s look deeper to the first two block:

1. Conv2DTranspose:

|  |
| --- |
| u6 = Conv2DTranspose (128, (2, 2), strides=(2, 2), padding='same') (c5) |

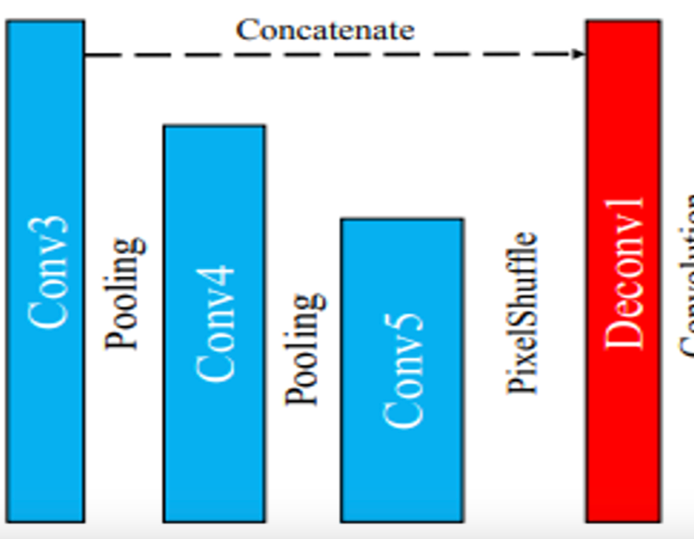
It is the opposite of Conv2D layer where Conv2D attempts to compress inputs, while Conv2DTranspose (upsampling) attempts to expand inputs. See figure (12)

  
Figure (12)

1. Concatenate:

|  |
| --- |
| u6 = concatenate ([u6, c4]) |

convolutional and pooling layers are first used to downsample the input images into feature maps, and then four deconvolutional layers are used to upsample back to original image size.the concatenate takes a list of layers as input, all the layers should be same size, and returns a single output that is a sequence of all the inputs.as shown in figure (13)

  
Figure (13)

So, the code for second step become:

|  |
| --- |
| u6 = Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same') (c5)  u6 = concatenate([u6, c4])  c6 = Conv2D(128, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (u6)  c6 = Dropout(0.2) (c6)  c6 = Conv2D(128, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (c6)  u7 = Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same') (c6)  u7 = concatenate([u7, c3])  c7 = Conv2D(64, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (u7)  c7 = Dropout(0.2) (c7)  c7 = Conv2D(64, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (c7)  u8 = Conv2DTranspose(32, (2, 2), strides=(2, 2), padding='same') (c7)  u8 = concatenate([u8, c2])  c8 = Conv2D(32, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (u8)  c8 = Dropout(0.1) (c8)  c8 = Conv2D(32, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (c8)  u9 = Conv2DTranspose(16, (2, 2), strides=(2, 2), padding='same') (c8)  u9 = concatenate([u9, c1], axis=3)  c9 = Conv2D(16, (3, 3), activation='elu', kernel\_initializer='he\_normal', padding='same') (u9)  c9 = Dropout(0.1) (c9) |

The ended size here is 400x400x16 , to make it the same size of input we add one extra Conv2D layer with 3x3 kernel size to end up with size 400x 400x 3 :

|  |
| --- |
| outputs = Conv2D(3, (3, 3), activation='relu', kernel\_initializer='he\_normal', padding='same') (c9) |

As seen in last layer we have 3 neurons so we have three outputs (image height, image width, channels)

Now our model represented as inputs and outputs

|  |
| --- |
| model = Model(inputs=[inputs], outputs=[outputs]) |

**Third Step:**

Next step now is to compile our model with suitable optimizer and learning rate :

TYPES OF OPTIMIZERS :

* Gradient Descent
* Stochastic Gradient Descent
* Adagrad
* Adadelta
* RMSprop
* Adam

Adam optimizer is used in this model which is algorithm that also works on the method of calculating adaptive learning rates for each parameter in each iteration. It uses a combination of Gradient Descent with Momentum and RMSprop to specify parameter values.

Now that the model is ready, it's time to work on the data

**3. Data Proccising**

The Main difference between mode 1,2,3 is the data but all of them sharing the same model proccese asn same data preproccicing

**While Model 4:**

It is a trained model you just choose content image and the style image , the result will appear immediately

**4.1 Models outputs**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model | X\_train | Y\_train | X\_test | Y\_predict |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 | \_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_ | Content image:    Style image: |  |

**4.2 Output loss diagram**

There is 2 loss :

Training loss and validation loss ,as shown in figure (14) the loss dicrease in each epoch and validation loss too , but Because there is not enough time validation steps is just 200 .

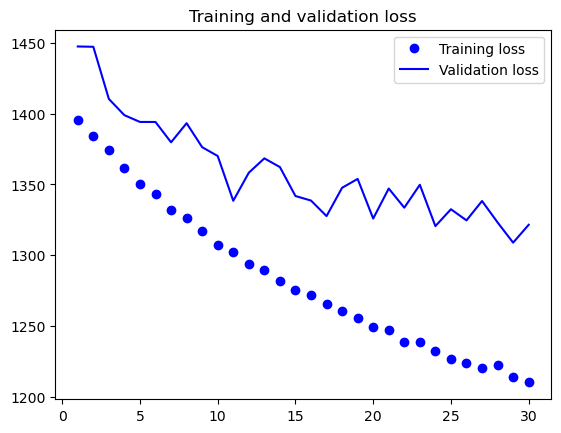


Figure (14)

**5.Data Sources**

**Model 1:** [**https://www.kaggle.com/code/nathannguyendev/face2comic/data?select=face2comics\_v2.0.0\_by\_Sxela**](https://www.kaggle.com/code/nathannguyendev/face2comic/data?select=face2comics_v2.0.0_by_Sxela)

**Model 2:**

[**https://www.kaggle.com/datasets/defileroff/comic-faces-paired-synthetic**](https://www.kaggle.com/datasets/defileroff/comic-faces-paired-synthetic)

**Model 3:**

[**https://www.kaggle.com/code/vbookshelf/art-by-ai-selfie-painter-tfjs-web-app/notebook**](https://www.kaggle.com/code/vbookshelf/art-by-ai-selfie-painter-tfjs-web-app/notebook)

**Model 4:**

[**https://tfhub.dev/google/magenta/arbitrary-image-stylization-v1-256/2**](https://tfhub.dev/google/magenta/arbitrary-image-stylization-v1-256/2)

**References**

[1] <https://python-course.eu/machine-learning/dropout-neural-networks-in-python.php>

[2] <https://closeheat.com/blog/elu-activation-function>

[3] <https://stackoverflow.com/questions/52578950/causal-padding-in-keras>

[4] <https://www.numpyninja.com/post/how-padding-helps-in-cnn>

[5] <https://keras.io/api/layers/convolution_layers/convolution2d/>

[6] <https://www.v7labs.com/blog/neural-networks-activation-functions>

[7] <https://iq.opengenus.org/conv2d-in-tf/>

[8] <https://medium.datadriveninvestor.com/an-overview-on-u-net-architecture-d6caabf7caa4>

[9] <https://towardsdatascience.com/u-net-b229b32b4a71>

[10] <https://www.geeksforgeeks.org/keras-conv2d-class/>

[11] <https://www.britannica.com/art/caricature-and-cartoon>

[12] <https://heartbeat.comet.ml/deep-learning-for-image-segmentation-u-net-architecture-ff17f6e4c1cf>

[13] <https://medium.com/@bdhuma/which-pooling-method-is-better-maxpooling-vs-minpooling-vs-average-pooling-95fb03f45a9>

[14] <https://stackoverflow.com/questions/68976745/in-keras-what-is-the-difference-between-conv2dtranspose-and-conv2d>

[15] <https://medium.com/@marsxiang/convolutions-transposed-and-deconvolution-6430c358a5b6>

10