12.04.22

Computer Vision

Facial Recognition / alignment

Candidate number :

A picture containing text, athletic game, sport

Description automatically generated

# **Chapter One**

# **Facial Alignment System**

## **Summary & Justification of Facial Alignment System**

To start this off, I would like to explain my choices of preprocessing. Since we are working with 244x244 images, I have made sure all the training points are in a range of between 0-244. This was done by using the np.clip operator.



Another form of preprocessing I have used is altering the pixel intensity by using the ImageDataGenerator function. I started off by creating a new copy of the initial image and then, expanding the proportions of that image. I then used a random dot operator to check whether to increase or decrease the brightness of the image, and then return the new processed image.

Text

Description automatically generated

In this facial alignment system, I have used CNN’s (Convolutional Neural Networks). A convolutional neural network (CNN) is a form of artificial neural network that is especially intended to analyze pixel input and is used in image recognition and processing *(What is convolutional neural network? - Definition from WhatIs.com, 2022).* In this system, we have a total of 2,811 images that are 244x244 pixels. In this system, I have used a total of five convolutional layers. There is no specific reason as to why I have chosen to work with five layers other than it was what produced the best results.

Table

Description automatically generated with medium confidence

**Figure 1.0 Convolutional Layers Used**

These convolutional layers use filters (kernels) and channels of filters to unsheathe specific features of an image. A kernel is a matrix that is slid over the picture and multiplied with the input to improve the output in a desired way *(Ganesh, 2022).* Thus, for each convolutional layer, I have used different values of either the filters or kernels, or both. Convolutional neural networks use a filter to summarize the existence of observed features in an input *(Brownlee, 2022).* These parameters are the number of filters that the model will learn*.*  Before explaining the reason I have used values such as 8, 16, 32, 64, and 128, I would like to explain the logic behind why the value of the filter plays a significant role.

For instance, let’s take the first convolutional layer I have used. The filter being used is 8. This provides the model with 8 various ways of extracting features from an input, or many distinct ways of both "learning to see" and "seeing" the input data after training *(Brownlee, 2022).*  The ‘8’ parameter in the first convolutional layer represents the number of units and the ‘5,5’ parameters represent the window size.

In my first convolutional layer, I have specified the input shape as it allows the convolutional neural network to use it as a building block. After that layer, each time the syntax ‘model.add’ is used, The previous layer serves as the following layer's input *(Ihare, 2022).*

Figure 1.0 demonstrates that in my code, the values used in the window size parameter in all five convolutions are odd. This is as all the preceding layer pixels would be symmetrical around the output pixel with an odd-sized filter. We'll have to account for distortions across the layers if we don't have this symmetry *(Ihare, 2022).*  Because we are working with images that are larger than 128x128, I have started with (5,5) kernels rather than (3x3), however, used (3x3) and (1,1) sized kernels in the convolutional layers afterwards. This aids the machine in the learning of bigger spatial filters and the reduction of volume size *(Rosebrock, 2022).* Starting with a higher kernel size such as (5,5) also helps in learning large features and then reducing special dimensions (*Rosebrock, 2022).*

As the number of convolutional layers increases, I also increase the number of filters that the network will learn also increases. This is due to max pooling which will be explained later. Maxpooling reduces the spatial dimensions and as the spatial dimensions decrease with each layer, the number of filters learnt is increased. Doing it this way allowed me to see the best results and hence, I stook with it.

Maxpooling, as mentioned previously, decreases picture dimensionality by lowering the number of pixels in the preceding convolutional layer's output *(Max Pooling in Convolutional Neural Networks explained, 2022).* When using maxpooling, we use a ‘nxn’ filter. In my case, all of my nxn values were 2x2. In other words, I have used a 2x2 filter in all my maxpoolings. This means that we take the first 2 x 2 region from the convolutional output and calculate the maximum value from each value in the 2 x 2 block. This value is saved in the output channel, which contains the entire result of the max pooling operation *(Max Pooling in Convolutional Neural Networks explained, 2022). Th*is procedure is repeated over the whole image, and when we're done, we're left with the new image representation, the output channel *(Max Pooling in Convolutional Neural Networks explained, 2022).*

**Benefits of Using MaxPooling:**

* ***Computational Load Reduction:*** Because max pooling decreases the resolution of a convolutional layer's supplied output, the network will be looking at bigger portions of the picture at a time in the future, lowering the number of parameters in the network and therefore reducing computing burden *(Max Pooling in Convolutional Neural Networks explained, 2022).*
* ***Overfitting Reduction***: Overfitting may be reduced by using maximum pooling. Our network will be seeking to extract certain specific characteristics for a given image, which is why max pooling works *(Max Pooling in Convolutional Neural Networks explained, 2022).*

## **Flowchart Summary of my Code:**

ON THE NEXT PAGE

Diagram

Description automatically generated with medium confidence

# **Chapter Two**

# **Experiment Results**

## **2.1 Training & Epochs**

In this design, we were given many 244x244 example images. The purpose of the project was to train the machine into predicting the testing images as accurately as possible. I used train\_test\_split to divide our data into training and assessment sets. The number of layers, values of filters / kernels and number of epochs all play a role in the output of the results. Epochs essentially act as ‘generations’ for the machine. The more epochs included, the greater number of times that the machine will learn about the algorithm and potentially adapt to producing slightly more accurate results. Each sample in the training dataset has had the chance to change the internal model parameters once each epoch *(Brownlee, 2022).*

In my code, I have used a total of 50 epochs. As I only have five convolutional layers, the epochs are generated more swiftly.

It should be noted that, as the number of epochs increase, the accuracy does too, as shown in this graph below.

Chart

Description automatically generated

**Figure 1.1 Epoch vs Accuracy (50 epochs)**

## The number of classifications a model successfully predicts divided by the total number of predictions is known as model accuracy. This is just a method of evaluating a model’s performance but it not being the only one. *(The Value of Model Accuracy, 2022).*

Using 50 epochs gives me an accuracy of 0.98 and a loss of 45.



## **2.2 Facial Alignment Quantitative Results**

A vital factor that affected the results were the number of epochs. In my design, the high number of epochs created more accurate results with lesser losses. In this section I will be altering the number of epochs and comparing the accuracy and results of the images. I will also alter the number of CNN layers used and see how that effects the output.

A picture containing diagram

Description automatically generatedA picture containing calendar

Description automatically generated

A picture containing application

Description automatically generatedGraphical user interface

Description automatically generated

The four pictures shown above shows my resulted output. These four pictures were specifically chosen as they are all colored differently and the people in these pictures have their head/faces angled in different ways. This is a good way to detect any errors by finding patterns. More examples of my face alignment results are below.

Graphical user interface, application

Description automatically generatedGraphical user interface

Description automatically generated with medium confidenceA picture containing graphical user interface

Description automatically generated

These were my results with 50 epochs and five CNN layers.

To achieve these results, I have used five layers, each filter / kernel and window values will be listed below.

**Layer 1:** Filter Value: 8, Window Values: (5,5)

**Layer 2:** Filter Value: 16, Window Values: (3,3)

**Layer 3:** Filter Value: 32, Window Values: (3,3)

**Layer 4:** Filter Value: 64, Window Values: (1,1)

**Layer 5:** Filter Value: 128, Window Values: (1,1)

All **MaxPooling** Layers had values of (2,2)

**Epochs**: 50

Before using these values, I used different window values. Instead of (1,1) I used (3,3), with 20 Epochs. Using these values decreased accuracy and impacted the output as seen below.

Chart, line chart

Description automatically generated

As seen here, doing so changed my accuracy from 0.98 to 0.94. This can also be seen as there are now more images that are not properly aligned.

Graphical user interface, application

Description automatically generatedApplication

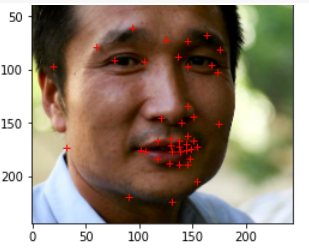
Description automatically generatedGraphical user interface, application

Description automatically generated

As seen in these photos, changing values of just two windows in two filters had a 4% difference in accuracy and this was seen obviously in the results.

## **2.3 Face Alignment Qualitative Examples**

Graphical user interface

Description automatically generated A picture containing text, indoor, looking

Description automatically generated

Graphical user interface

Description automatically generated with medium confidence Graphical user interface, application

Description automatically generated Graphical user interface, application

Description automatically generated

These are the ouputs of the 6 example images.

## **2.4 Points of Failure**

Some noticeable patterns that effect the alignment of the face that are worth mentioning:

* Direction / angle of face
* Size of face
* Facial features (symmetry, facial hair, etc.)

Graphical user interface, application

Description automatically generated

This lady turning right shows that the face angle is a liability in face alignment system.

Graphical user interface

Description automatically generated

This man giving a speech shows a good example of face size impacting the accuracy ofalignment.

Graphical user interface, application

Description automatically generated

The facial hair of this man interrupts alignment of the lips and is a good example to outline the point of failure.

## **2.5 Euclidean Distance:**

Created a histogram displaying predicted vs definite points.

## **2.6 Lip Modification System**

* For my lip modification system, I use the numpy copy operator in order to get an array copy of the image I selected. I then use the cv2 library with the .fillpolly operator and add parameters to it and set the value of pts to 22 so that the lips can be fully covered by the color chosen, (in my case, green). The .fillpolly operator is used to draw filled polygons over an image, such as rectangles, triangles, and pentagons *(Draw a filled polygon using the OpenCV function fillPoly() - GeeksforGeeks, 2022).* I then set my transparency variable to 0.7 to make it a strong and clear color on the lips. In the parameters, I use the image I need to modify the lips on, in this case, image 234, and use the RGB method to apply the green color (0, 255, 0). Below is what my lip modification result looks like.
* Graphical user interface, application

  Description automatically generated

# **References**

1. SearchEnterpriseAI. 2022. *What is convolutional neural network? - Definition from WhatIs.com*. [online] Available at: <https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network> [Accessed 10 May 2022].
2. Ganesh, P., 2022. *Types of Convolution Kernels : Simplified*. [online] Medium. Available at: <https://towardsdatascience.com/types-of-convolution-kernels-simplified-f040cb307c37> [Accessed 10 May 2022].
3. Brownlee, J., 2022. *How Do Convolutional Layers Work in Deep Learning Neural Networks?*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/> [Accessed 11 May 2022].
4. Brownlee, J., 2022. *How Do Convolutional Layers Work in Deep Learning Neural Networks?*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/> [Accessed 11 May 2022].
5. Ihare, A., 2022. *Significance of Kernel size*. [online] Medium. Available at: <https://medium.com/analytics-vidhya/significance-of-kernel-size-200d769aecb1> [Accessed 11 May 2022].
6. Rosebrock, A., 2022. *Keras Conv2D and Convolutional Layers - PyImageSearch*. [online] PyImageSearch. Available at: <https://pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/> [Accessed 11 May 2022].
7. Rosebrock, A., 2022. *Keras Conv2D and Convolutional Layers - PyImageSearch*. [online] PyImageSearch. Available at: <https://pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/> [Accessed 11 May 2022].
8. Deeplizard.com. 2022. *Max Pooling in Convolutional Neural Networks explained*. [online] Available at: <https://deeplizard.com/learn/video/ZjM\_XQa5s6s> [Accessed 11 May 2022].
9. Deeplizard.com. 2022. *Max Pooling in Convolutional Neural Networks explained*. [online] Available at: <https://deeplizard.com/learn/video/ZjM\_XQa5s6s> [Accessed 11 May 2022].
10. Deeplizard.com. 2022. *Max Pooling in Convolutional Neural Networks explained*. [online] Available at: <https://deeplizard.com/learn/video/ZjM\_XQa5s6s> [Accessed 11 May 2022].
11. Deeplizard.com. 2022. *Max Pooling in Convolutional Neural Networks explained*. [online] Available at: <https://deeplizard.com/learn/video/ZjM\_XQa5s6s> [Accessed 11 May 2022].
12. Deeplizard.com. 2022. *Max Pooling in Convolutional Neural Networks explained*. [online] Available at: <https://deeplizard.com/learn/video/ZjM\_XQa5s6s> [Accessed 11 May 2022].
13. Brownlee, J., 2022. *How Do Convolutional Layers Work in Deep Learning Neural Networks?*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/> [Accessed 11 May 2022].
14. DataRobot AI Cloud. 2022. *The Value of Model Accuracy*. [online] Available at: <https://www.datarobot.com/blog/the-value-of-model-accuracy/> [Accessed 11 May 2022].
15. GeeksforGeeks. 2022. *Draw a filled polygon using the OpenCV function fillPoly() - GeeksforGeeks*. [online] Available at: <https://www.geeksforgeeks.org/draw-a-filled-polygon-using-the-opencv-function-fillpoly/#:~:text=fillPoly()%20function%20of%20OpenCV,endpoints%20of%20Polygon%20and%20color.> [Accessed 11 May 2022].