UTILIZING MULTI-TASK CASCADED CONVOLUTIONAL NETWORKS AND RESNET-50 FOR FACE IDENTIFICATION TASKS

by

Chuanyang Cai

B.Sc., The University of British Columbia, 2018

PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER SCIENCE

UNIVERSITY OF NORTHERN BRITISH COLUMBIA

December, 2021

Abstract

In recent years, computer-animated characters have been designed more and more vivid and lifelike, and many of them are extremely similar to real people. Due to the high degree of similarity, some classical face recognition models are mixed up in the face identification process. For example, FaceNet model matches cartoon facial images with their similar real faces. In this case, some people may try to cheat by using virtual faces when they are identified by face recognition systems. To address this problem, this paper proposes an integrated approach that utilizes Multi-task Cascaded Convolutional Networks (MTCNN) and Resnet-50 models for the classification of real and cartoon faces (or virtual faces) of an input image before the face identification task. Our experiments show that the proposed integrated approach achieves better results on face identification tasks compared to some classical face recognition models that accomplish the tasks directly.

Contents

List of Figures

Acknowledgments

I would like to acknowledge and give my warmest thanks to my supervisor, Professor Chen, who made this work possible. His guidance and advice carried me through all stages of writing the project. I would also like to thank the members of my committee, Professor Jiang and Professor Sui, for their excellent comments and suggestions.

Chapter 1

Introduction

1.1: Overview

In the field of computer vision, face recognition technology is one of the popular applications that has been studied for more than 50 years [1]. Over the past few decades, scientists have continued to improve face recognition techniques. American researcher Bledsoe first used a semi-automatic method for face recognition. Later, some scientists tried to use support vector machines (SVMs) or random forests to improve the efficiency and accuracy of face recognition [2]. Currently, the combination of deep learning methods and digital image processing techniques has been widely used to replace previous methods due to their ability to achieve higher accuracy in face recognition tasks [1]. One of the most popular techniques in the field of deep learning is the convolutional neural network (CNN), which is the main method applied in this project.

Recently, face recognition technology has been widely used around the world to solve different problems. A typical example is that this technology can be used in surveillance systems to help people automatically locate a specific individual face in a photo or video, which is more efficient and accurate than a manual search. Before starting this project, it is necessary to define face detection and face recognition. In fact, these two concepts are

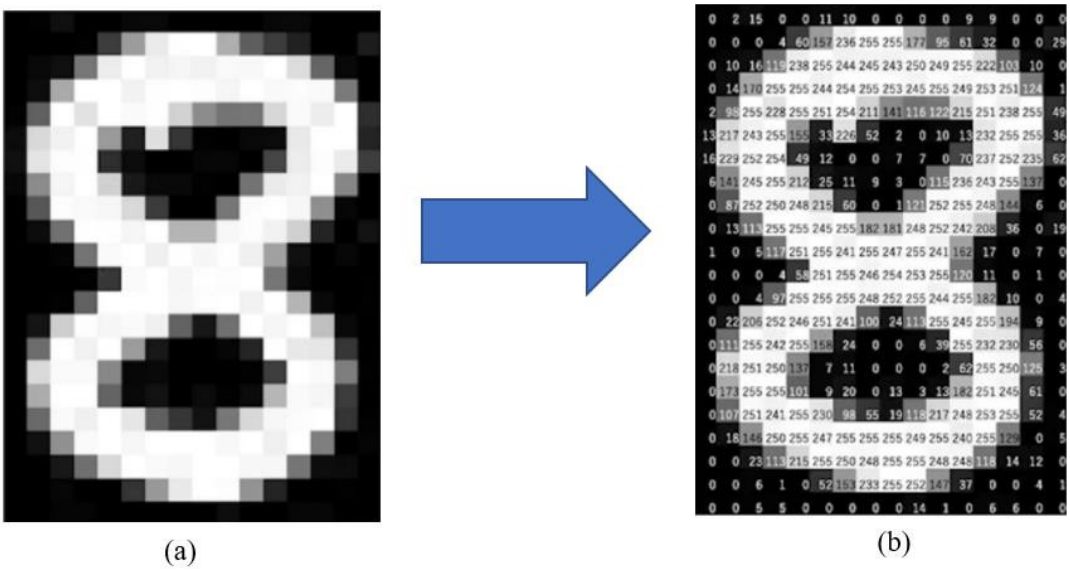
different. The former is always the first step of face recognition and is mainly used to find every face in an image. The algorithms for these two concepts also different. For example, in this project, we use the Multi-task Cascaded Convolutional Networks (MTCNN) for face detection tasks. While for face recognition tasks, we mainly utilize FaceNet algorithm. To make the whole process clearer, we summarize the face recognition technique in the next paragraph.

Generally speaking, the face recognition technique can be divided into three stages.

1) The first stage is known as face detection and is used to locate a face from the image. In this stage, the detector will use a bounding box to mark each face in the image. Based on the bounding box, the detector will also return the coordinates of each face.
2) In the second stage, after obtaining the position of the face, we must extract facial features from the face and then convert them into a feature vector by image processing techniques. This stage is generally known as facial feature extraction.
3) In the third stage, we must use the facial feature vector computed from the second stage, so that it can be compared with other feature vectors. For each comparison, we will get a similarity score which is then used to express the probability that they belong to the same face. Therefore, the third stage is also known as face matching or identification.

1.1.1: Digital Image Processing

Before introducing the convolutional neural networks, it is necessary to know some background knowledge about digital image processing techniques. Digital image processing is a method used to convert an image into a two-dimensional function, $f(x, y)$, where x and y refer to the plane coordinates [3]. Within the coordinate system, the value of f at any pair of coordinates (x, y) is called the gray level of the image at that point. Each element has a specific location and a gray level, and these elements are also called pixels. The value of each pixel always ranges from 0 to 255 (i.e., black to white). In the case that an image contains a finite number of pixels, it is called a digital image [3].



(a) (b)

Figure 1: Figure 1(a) is a grayscale image of the number eight, and 1(b) has more specific brightness information for each pixel [18].

For example, Figure 1(a) represents a grayscale image of the digital number 8. The grayscale image implies an image that contains only luminance information [18]. Generally,

in a grayscale image, the bright parts (i.e., close to 255) are white; while the dark parts (i.e., close to 0) are black. In Figure 1(a), we can find that the number 8 is white, while the background of this image is black. In addition, there are many small white and black squares in this image, which are called pixels. In Figure 1(b), for the height(y) of this image, we can count that there are 24 pixels. And for the width of this image, there are 16 pixels. Therefore, this image can be transformed into a two-dimensional function $f(16, 24)$. In addition, it can be found that each pixel has a value between 0 and 255. For each pixel value, the white pixels are close to 255; while the black pixels are close to 0. By using a computer and some algorithms, the obtained digital image can be processed. In the field of computer vision, one of the typical algorithms is known as convolutional neural network, which will be described in the next paragraphs.

1.1.2: Convolutional Neural Network (CNN)

The concept of the convolutional neural network was first introduced by a computer scientist named Yann LeCun in the 1980s [4]. At that time, an early version of CNN was named LeNet and could be used to recognize handwritten digits [4]. However, CNN technology was not widely used due to the limitation of computing hardware for training network models. It was not until 2012 that Alex Krizhevsky used graphics processing units (GPUs) to train the model and designed the AlexNet, which achieved a top-five error of 15.3% in the ImageNet Large-Scale Visual Recognition Challenge (LSVRC) [4]. Since

then, more and more scientists have started to revisit the CNNs and have improved them significantly over the past decades.

With the rapid development of deep learning, convolutional neural networks have become the main method for face recognition. To achieve better performance, computer scientists have proposed a variety of convolution neural network architectures. For example, a Facebook research group created a famous CNN architecture, DeepFace, for recognizing human faces in digital images [13]. The researchers used more than 4.4 million facial images to train their CNN model. Finally, by using the DeppFace model, the accuracy of face recognition reached about 97.35% [13]. Another famous face recognition model was built by a research group at the Chinese University of Hong Kong called Deep Identification-Verification Features (DeepID2). This model allows face recognition accuracy to reach a higher value of about 99.15% [14].

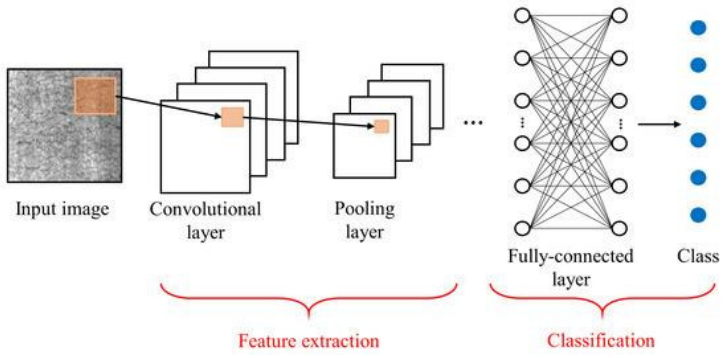


Figure 2: This diagram illustrates a typical architecture of the convolutional neural network [19].

In general, the main architecture of CNN model is divided into two parts, as shown in Figure 2. The first part is the feature extraction layer and the other is called the

classification layer.

1) In the feature extractor, many extraction layers are used to receive the outputs from their immediate previous layers as the inputs and send their outputs as the inputs to the next layers.
2) The classification layer, also known as the fully connected layer, is used to compute the score for each category from the feature extraction layer. In this layer, a technique called stochastic gradient descent (SGD) is used to optimize the objective function. Then, a backward propagation method is applied to update the input weights. Finally, both methods are executed recursively until the losses are minimized and the final outputs are obtained.

In addition to the two components mentioned above, another essential component of the CNN model, namely activation function, is also included in the classification layer. One of the most important reasons to explain the necessity of the activation function for the CNN model is that it can make the network become non-linear. In other words, the inclusion of the activation function can make the CNN network more easily adapt to some non-linear models. In the field of deep learning, there are some commonly used activation functions. Among them, the ReLU function is often used to solve the problem of gradient vanishing. For the binary classification task, we usually use the Sigmoid function as the solution. And for the muti-class classification task, the Softmax function is preferred.

1.2: The Inspiration and Motivation

In the previous section, some basic knowledge about digital image processing and CNN architecture has been presented. In this section, we will briefly describe the inspiration and motivation that make me do this project.

1.2.1: Computer Animation and Object Detection

Today, watching computer animation is a popular hobby for most teenagers and adults around the world. One of the major reasons is that some cartoon characters, created by computers, are becoming more and more vivid and realistic. For example, on the left side of Figure 3, two cartoon characters are very similar to real people. Due to their high similarity, some object detection CNN models, such as Single Shot Multi-Box Detector (SSD), can also capture cartoon characters when they are used to detect real people (as shown in Figure 3). Therefore, we decided to give this model a new capability to classify real people and cartoon characters when it is used to detect them. However, due to the difficulty in collecting full-size images of people, we decided to focus on the collection of facial images instead.

Figure 3: The Object detection results by applying the SSD model (The SSD model can capture both real and carton people from an image).

1.2.2: Computer Animation and Face Identification

In recent years, the accuracy of the face recognition technology has been improved dramatically, and in 2020, the best face recognition system had an error rate of 0.08% [7]. However, we find that some classical face identification models may match cartoon faces with their similar real faces. For example, in Figure 4, we try to use the FaceNet model for an identification task. In this example, we choose a famous Chinese actor named Zhan Xiao and his cartoon facial image. Based on the matching name (as shown on the button of the blue bounding box), we can understand that the FaceNet model matches Zhan Xiao's cartoon face (i.e., the image on the right) with his real face (i.e., the 51st image in the database on the left). Therefore, this result demonstrated that the FaceNet model could match a cartoon face with a similar real person's face. From this experiment, it can be found that some cartoon or virtual faces can negatively affect the accuracy of the face recognition task. Therefore, we decided to propose an integrated approach to deal with this situation.

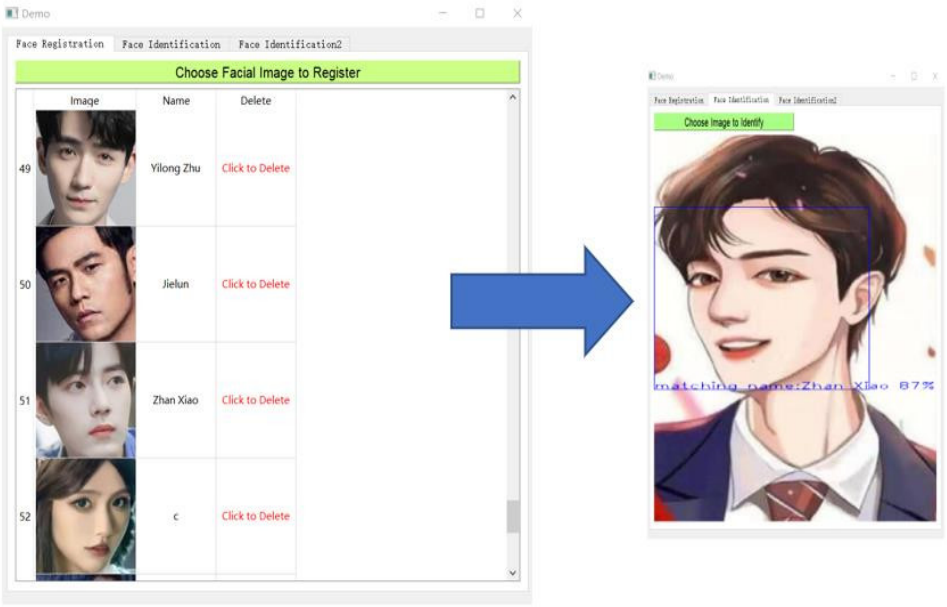


Figure 4: In this figure, the image on the left is a small dataset that is used to save some registered faces. The image on the right is the result of cartoon face identification by applying the FaceNet model.

In this paper, we mainly focus on the introduction of the deep learning methods of face detection and identification tasks. In the field of deep learning, there are a variety of CNN models used to solve the above two problems. In the literature review section, we will introduce one of the classical face-detection models named Multi-task Cascaded Convolutional Networks (MTCNN). This model was proposed by Kaipeng Zhang and his team members in 2016 [6]. To further improve its performance, we added another popular model, named Resnet-50, as a classifier. The final results show that the integration of the two network models can classify real and cartoon faces from the input images. For face identification tasks, we choose the FaceNet model to match faces, and it is also introduced in the literature review section.

1.3: Research Contribution

The main contribution of this project is the adoption of an integrated method to improve the accuracy of some face recognition models that are utilized in the special case (i.e., To detect and identify real and cartoon faces). In the face identification process, according to our experiments, some classical face recognition models cannot avoid matching a vivid cartoon or virtual designed face with its similar real personal face (shown in Figure 4). In order to deal with this situation, we use the proposed integrated approach to examine the input facial images and filter out each cartoon or virtual designed face before doing the face identification tasks. Figure 11 illustrates that better results can be obtained on the face identification tasks when using our proposed approach.

1.4: An Overview of the Rest of This Project

Chapter 2 briefly discusses several important models and theories that are helpful for this project. The first model is a face detection model named Multi-task Cascaded Convolutional Networks. And a classification model named ResNet-50. Finally, a face identification model named FaceNet.

Chapter 3 describes our proposed integrated approach and shows some relevant experimental results;

Chapter 4 discusses the main contribution of this project and two potential applications based on our proposed integrated approach;

Chapter 5 summarizes this project and concludes several future improvements.

Chapter 2

Literature Review

2.1: Multi-task Cascaded Convolutional Networks and Face Detection

In the paper [6], Kaipeng Zhang and his team members proposed a new framework utilizing the unified cascaded CNNs by multi-task learning (MTCNN), which integrates face alignment and detection for better performance on face detection. In addition, they proposed an efficient automatic online hard sample mining strategy to replace the manual sample selection in the learning process [6].

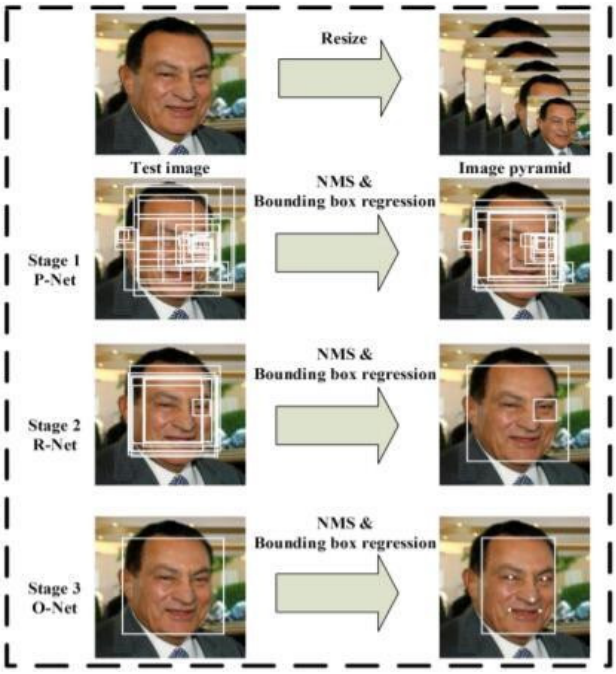


Figure 5: The pipeline of multi-task Cascaded Convolutional Networks [6]

This proposed model consists of three networks. Before the input of the image to the three-stage cascaded networks, the image is resized to different scale to create an image pyramid (i.e., the first step in Figure 5). The resize-factor generally ranges from 0.70-0.80, which is because that the value of this range can avoid the missing of some small faces in the detection process. After obtaining an image pyramid, it will be inputted to the first network named Proposal Network (P-Net). As a full convolutional network, this network is used to obtain the candidate windows and their bounding box regression vectors. After obtaining these candidate windows, they use the non-maximum suppression (NMS) to for the reduction of some overlapped candidates. After that, the remaining candidates will be inputted to the second network called Refine Network (R-Net). This network is used to filter some false candidates from the first stage. In this stage, they also use the NMS to merge some highly overlapped candidates. In the third stage, the processed images will be inputted to the network named Output Network (O-Net). Though this network is similar to the second one, it mainly focuses on obtaining more details of the faces in the image. At last, the position of the five facial landmarks will be outputted from the network.

Due to the fact that the MTCNN model has three independent networks, the training process is also slightly different. Initially, they have to utilize the data to train the P-Net. When they obtain the output trained model from the P-Net, they can input it to the next R-Net for training. At last, they have to use the output from the R-Net to train the O-Net. In short, the training output of the previous network acts as the training input of the following

network. In each network, it has three different tasks, i.e., face classification, bounding box regression, and facial landmark localization. For each task, the authors utilize different loss function for optimization.

1) In the face classification task, they use the cross-entropy loss for each sample x_i:

$$L_i^{det} = -(y_i^{det} \log(p_i) + (1 - y_i^{det})(1 - \log(p_i)),$$

where p_i refers to the probability used to indicate whether the sample is a face or not, and $y_i^{det} \epsilon \{0,1\}$ denotes the ground-truth label.

2) In the bounding box regression task, they use the Euclidean loss for each sample x_i:

$$L_i^{box} = \left\| \hat{y}_i^{box} - y_i^{box} \right\|_2^2,$$

where $\hat{y}_i^{box}$ refers to the regression target obtained from the network, and y_i^{box} denotes the ground-truth coordinate.

3) In the facial landmark localization task, they also use the Euclidean loss for each sample x_i:

$$L_i^{landmark} = \left\| \hat{y}_i^{landmark} - y_i^{landmark} \right\|_2^2,$$

where $\hat{y}_i^{landmark}$ refers to the facial landmark's coordinate obtained from the network, and $y_i^{landmark}$ represents the ground-truth coordinate.

2.2: Residual Network and Classification

In the deep learning field, the depth of a neural network has a significant influence on the accuracy of the final outputs. With the development of computational hardware, many

researchers try to deepen their network architecture. However, they also realized that when the depth of a network approaches a particular threshold value, the output accuracy of this network will decrease. At the beginning, some scientists believe that the decrease of accuracy due to the vanishing gradients would occur when the depth of the network architecture reached a particular point. In 2015, in the paper [5], Kaiming He and his team members did not believe that the problem of vanishing gradients was the key factor of the decrease of accuracy as the deepening of the network, which is because that this problem had been largely addressed by normalized initialization [8] [9] [10] [11] and intermediate normalization layers [12]. According to the experiment, they believe that the reason why deeper network shows a worse performance is due to the degradation (i.e., deeper neural networks are more difficult to train). To solve this problem, they proposed a residual learning framework. For the residual neural network, there are several different architectures. While in this paper, we only focus on the introduction of the 50-layer residual network (i.e., Resnet-50).

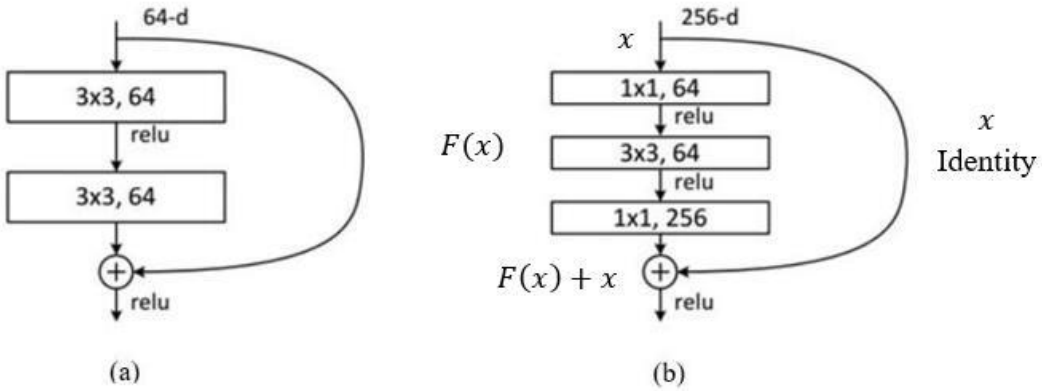


Figure 6: The two residual blocks of the Resnet architecture [5].

In the paper [5], the authors introduce two residual blocks (as shown in Figure 6) used for different depth of networks. In Figure 6(a), the residual block is generally used in the shallow networks of 18-layers and 34-layers, respectively. For the residual block in Figure 6(b), it is used in the deeper networks (i.e., 50-layers, 101-layers and 152-layers), and we will only consider this block as the key architecture in the Resnet-50 model.

In general, for a stacked layers network, assume the input is x, the output mapping of this stacked layers will be denoted as $H(x)$. In the residual network, the authors propose a new idea that they can divide the input processing into two parts. In Figure 6(b), the input of this block is x, and there are two mappings can be used to get its outputs. One is denoted as $F(x)$ which represents non-linear mapping, and the other is an identity mapping denoted as x. At last, they will add the results of nonlinearity and identity mappings together to recast the original one (i.e., $H(x)$) into $F(x) + x$ (i.e., $H(x) = F(x) + x$). Therefore, they can also be written as $F(x) = H(x) - x$ for the non-linear mapping called residual mapping. According to their experiments, they finally find that it is easier to optimize the residual mapping than to optimize the original mapping. The authors use the ImageNet dataset for the evaluation of the residual network, and the ensemble of this network achieves 3.57% error on the ImageNet test set. This error rate was also the lowest one in the ILSVRC 2015 classification task.

2.3: FaceNet and Face Identification

In the paper [15], Google proposes a new face recognition model, called FaceNet, to improve the efficiency when implementing face identification at scale. The FaceNet model uses the Inception network, a popular convolutional neural network architecture in the computer vision field, to process the input images [16]. Each facial image will be mapped to a 128-dimensional byte feature vector by using the Inception network. After that, the authors add the L_2 normalization and Euclidean embedding for each mapping vector, indicating that an image x will be mapped into a 128-dimensional Euclidean space, and its embedding can be represented as $f(x) \epsilon R^{128}$. The overall process is shown in Figure 7.

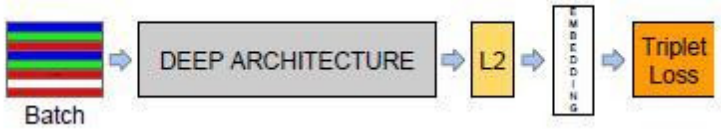


Figure 7: The model structure of FaceNet [15].

After obtaining the embedding $f(x)$, the main task is to optimize it. In their project, they use the Triplet Loss to minimize the distance between the two facial vectors. To understand the Triplet Loss, the authors define three vectors in the Euclidean Space, as shown in Figure 8. The anchor and positive vectors show the same identity, and the negative vector has a different identity. After learning, they hope to make the squared distance between the anchor and positive vectors small and that between the anchor and negative vectors large.

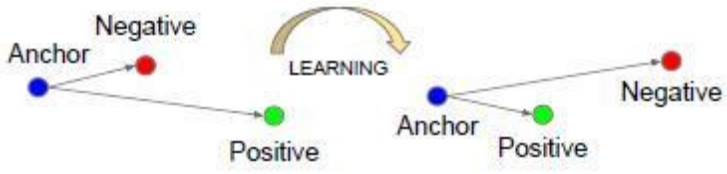


Figure 8: The triplet loss optimization of the FaceNet model [15].

In the paper [15], the authors also introduce the calculation of the Triplet Loss Optimization. They define the anchor image as x_i^a, the positive image as x_i^p, and the negative image as x_i^n. Then they transform Figure 8 into the mathematic equation as follows:

$$\left\|f(x_i^a) - f(x_i^p)\right\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2,$$

where α refers to a margin that is enforced between positive and negative pairs. If there are N training images in the dataset, the overall Triplet Loss (L) refers to the summation of each loss:

$$L = \sum_i^N \left\|f(x_i^a) - f(x_i^p)\right\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2$$

At last, the authors use the Labeled Faces in the Wild (LFW) dataset for the evaluation of the FaceNet model, and their system achieves the highest accuracy (99.63%) of face recognition.

Chapter 3

Methodology and Experiments

In our project, we integrate the three models to accomplish face recognition. The only difference compared with the traditional face recognition process lies in the fact that we will filter out cartoon facial images before inputting them to the face identification stage. According to our experiments, our integrated method exhibits better performance on the identification of cartoon facial images. The overall process can be divided into three stages: collecting data, creating a classifier, and setting up a small database.

3.1: Collect Facial Images and Set Up Datasets

In the first stage, we mainly focus on the collection of virtual (or cartoon) and real facial images from websites to create our datasets. Due to the fact that there are a variety of factors (such as facial expression, position, orientation, and so on) influencing the accuracy of face detection, we only collect the frontal faces in this project. Moreover, some cartoon faces cannot be detected by using the MTCNN model, which is not sensitive to capturing the 2-dimensional animated face from images. Therefore, we tend to choose more 3-dimensional cartoon faces to build our datasets. To filter out some useless factors (e.g., background colors, other objects, and so on) from the images, we decided to use the

MTCNN model to crop each of them so that we can get the faces only from images. The cropped images are divided and stored in two folders (i.e., train and test) as our datasets. We also create cartoon and real folders for train dataset (same as the test dataset) to save the collected images.

In our project, we manually collect 500 cartoon facial images and 500 real facial images for our datasets. For the 500 cartoon faces (same as the 500 real faces), we randomly chose 450 images for the training dataset and 50 images for the test dataset.

3.2: Create the Cartoon and Real Faces Classifier

In the second stage, the main task for us is to utilize the datasets to train the model, which can be used to classify real and cartoon faces of an image. In general, it requires a lot of data to obtain a good-performance model (for example, the amount of the required data is far more than 1000 images) and time for training by the convolutional neural networks. In that case, to accelerate the training speed and improve the accuracy of classification, we decided to use transfer learning which is a popular technique in the deep learning field. In addition, to increase the diversity of our training datasets, we utilize the data augmentation technique for the processing of each image. For example, we use random brightness, random horizontal flip, and other methods to process each input image, and these methods can make some tiny changes for each original image so that the CNNs will regard it as a

distinct image.

In this project, for our binary classification task (i.e., cartoon and real faces), we have downloaded the Resnet-50 pre-trained model from the official website as our basic model (i.e., the transfer learning). Before the training process, we also have to adjust the final outputs of the Resnet-50 model into two categorical outputs. After that, we can utilize the datasets built in the first stage to train the pre-trained model. At the completion of the training, we can get our model available to be used to classify real and cartoon faces from an image. Figure 9(b) shows the performance of our trained model, in which we use the red bounding box to mark cartoon face and the green bounding box to mark real face. Moreover, on the top of each bounding box, we record the probability of being cartoon or real face for each image. For example, on the top of the red bounding box, we can see that $Pr(cartoon) = 99\%$, indicating that this facial image has 99 percent of being a cartoon face.

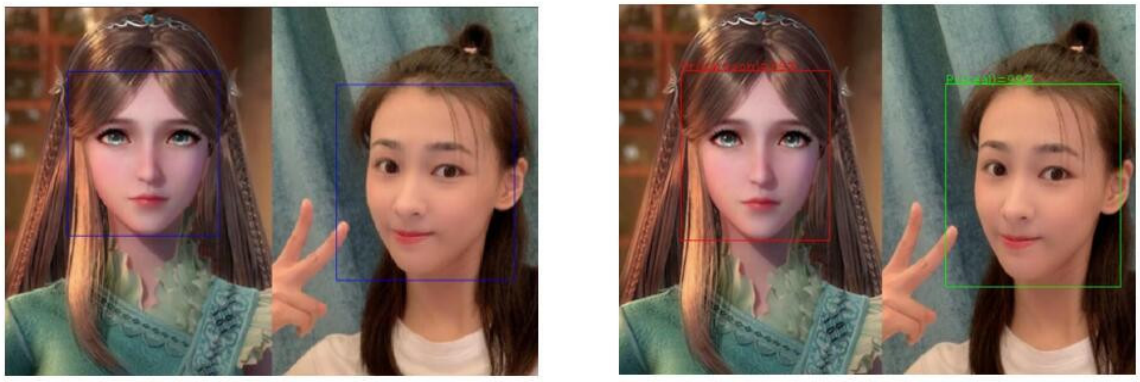


Figure 9: In Figure 9(a), we only use the MTCNN model to detect the faces, and this model can capture both real and cartoon faces. In Figure 9(b), we integrate the Resnet-50 model as a classifier to the MTCNN model so that they can classify real and cartoon faces from an image.

3.2.1: Cross Entropy Loss:

In the training process, the Cross Entropy Loss function is used for the evaluation of our model, which will be introduced in this section. In this project, we only have to classify real and cartoon faces, which means that our situation belongs to a two-classification task. Assuming that the probability of being a real face is p, then the probability of being a cartoon face will be $1-p$. The formula of Cross Entropy Loss in the binary situation is expressed as follows [17]:

$$L = \frac{1}{N}\sum_i L_i = \frac{1}{N}\sum_i -[y_i \cdot \log(p_i) + (1-y_i) \cdot \log(1-p_i)],$$

where y_i represents each input image, and i refers to a label (i.e., $i = 1$ means that the input image is a real face, while $i = 0$ means that the input image is a cartoon face). p_i denotes that the probability of the input image belongs to a real face.

3.2: Set Up Database for the Face Identification Tasks

In this stage, the first task for us is to set up a small database available for the storage of some real facial images. In this project, we use the Qt-Designer and SQLite to set up the user-interface, as shown in Figure 10. In the database, we have saved 100 real facial images. In this figure, we can view each image from the second column of the user interface, and the third column is used to record the name of each image.

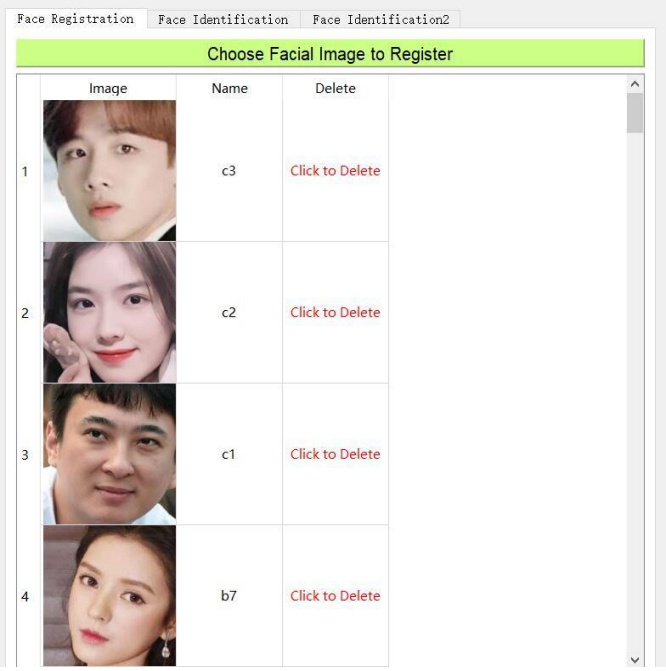


Figure 10: This is a database user interface which is created by the Qt-Designer, and all the images will be saved in the SQLite, which is a database connected with PyCharm.

After setting up the small database, we can use the FaceNet model for some face identification tasks. The identified image should be a new cartoon or real facial image which is not contained in the database. Previously, in Figure 4, the matching result has demonstrated that the FaceNet model is able to match a cartoon facial image with its similar real personal face. To avoid this situation, we use the integrated method to filter out the cartoon facial images before the face identification. The result of the application of our proposed method is shown in Figure 11(a), which demonstrates that cartoon images are not allowed to be used for face identifications. In addition, the system will pop out a notification window telling the operators that the given image belongs to a cartoon facial image. In Figure 11(b), if we choose a real facial image for identification, our proposed

method will show the matching rate and the matched name on the bottom of the bounding box. In the example, the given image is matched with a registered face named Qing Yan at the matching rate of 81%.

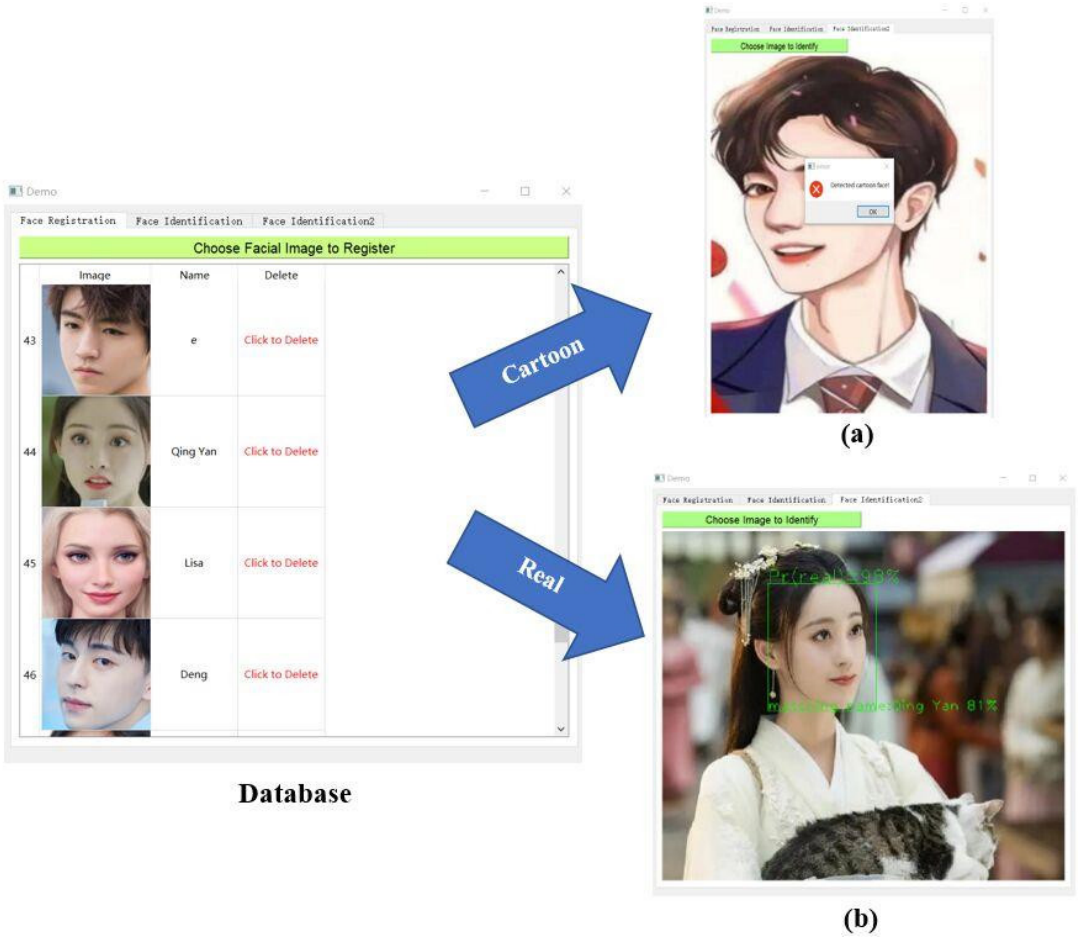


Figure 11: The image (a) shows a cartoon face identified result which is obtained by utilizing the integrated method, and Image (b) is a real face identified result.

Chapter 4

Research Contributions and Potential Applications

As mentioned in the end of Chapter 1, the main research contribution of this project is that we have proposed an integrated method applicable in the face recognition system. For this integrated method, we add the Resnet-50 as a classifier to the traditional face detection algorithm (i.e., MTCNN model) to make it available for the classification of real and cartoon faces from input images. After the classification of the real and cartoon faces, the face identification model will obtain better performance on the face recognition system. The next paragraph will summarize the entire process of our integrated method.

Since there are variety of cartoon facial images similar to real faces on the websites, some face recognition systems may match a cartoon facial image with its similar real personal face. In this case, some people may use their created cartoon facial images for cheating when they are going through face recognition systems. Therefore, some cartoon facial images may exhibit negative influences on the accuracy of face identification tasks. In view of this, we can use our proposed integrated method to classify the real and cartoon faces before starting the face identification tasks. Figure 12 is a flow diagram which summarizes the process of our proposed face recognition system. In this system, each input image will be checked by a classifier (i.e., our proposed method) first. In the case that the

image belongs to the cartoon type, it will not be allowed to be used in the face identification task. While if the image belongs to the real type, it will be sent to the next step to go through the face identification step. For this improved face recognition system, we initially utilized our proposed method to filter out the cartoon or virtual facial images from the input images. After that, there will be no cartoon images influencing the face identification process in this manner to make the whole system available for more precise results.

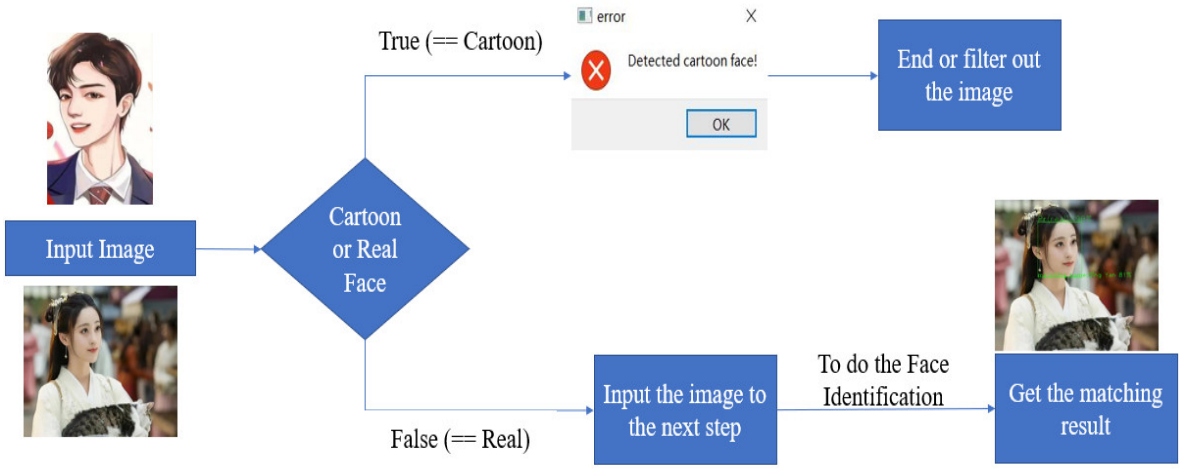


Figure 12: The flow diagram of the face recognition process by applying our proposed method.

Moreover, we would like to introduce two potential applications related to our integrated method. The following parts of this section will give more concrete descriptions.

4.1: The Classification of Image

The first potential application is that we can use our proposed method for automatic image classification. Today, most teenagers are fond of collecting different kinds of images by using their laptops. These images may contain cartoon characters' screenshots, real personal selfies, landscape images and so on. Consequently, it will be time-consuming for them to divide these images into different folders manually. To solve this problem, they can use our proposed method to automatically create a folder that only contains real people's selfies or cartoon characters' screenshots. Therefore, this potential application can help people to manage and check images on their laptops more easily.

4.2: The Photo Transmission Between Smartphone and Laptop

Another potential application is that we can use the integrated method as a mobile phone album classifier to help us filter the photos. Today, many teenagers and adults enjoy watching animations, and most of them get used to collecting some screenshots of the animation characters in their smartphones. Therefore, their smartphone albums will contain both real people and animated characters' photos at the same time. If they are going to import a particular type of photo (For example, they only want to import real people's photos.) from the smartphones to their laptops. In that case, they may have to manually

filter out all the cartoon images, which is quite time-consuming. In this case, our proposed method can be applied to the image transmission process, thereby helping people to filter out all cartoon photos automatically. Figure 13 summarizes the above description to make it more concrete.

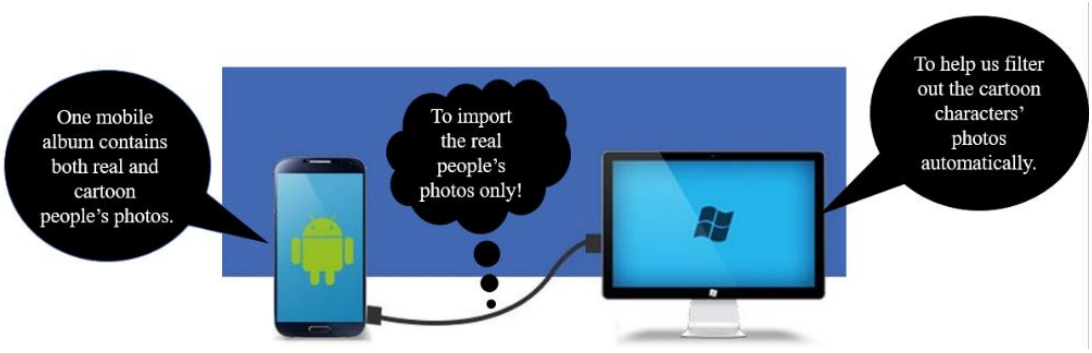


Figure 13: The process of image transmission by applying our proposed method.

Chapter 5

Conclusion and Future Work

5.1: Conclusion

Recently, the face recognition system has become one of the most useful technologies in the world. It can not only help people to automate some tedious tasks but also achieve more accurate results on some face recognition tasks that contain a large amount of data. In addition, in this paper, we propose two potential applications for our proposed face recognition method. For example, the technique can potentially be used in image management systems to help people classify their images. In addition, for the photo transmission process, people can use the integrated approach to automatically import one type of photos (i.e., cartoon or real type) from their smartphones to their laptops.

In general, face recognition technology has been widely used in different fields. However, some face recognition models still need to be improved continuously in the future. In this project, we proposed an integrated approach to face recognition task using MTCNN model and Resnet-50. From the experimental results, we can see that our integrated approach is able to classify real and cartoon faces from images. Moreover, our integrated method outperforms some traditional methods in face recognition systems when there are both real and cartoon faces.

In addition to giving the face recognition model the ability to classify both real and cartoon faces. In this project, we also found an interesting condition that should be improved in the future. Specifically, this condition occurred when we tested some animal images with our proposed integrated approach. In Figure 14, we can see that the input picture is a facial image of a monkey. However, the experimental result shows that our integrated model considers 92% of the given figure as a cartoon facial image, which is a completely wrong result made by our proposed approach.

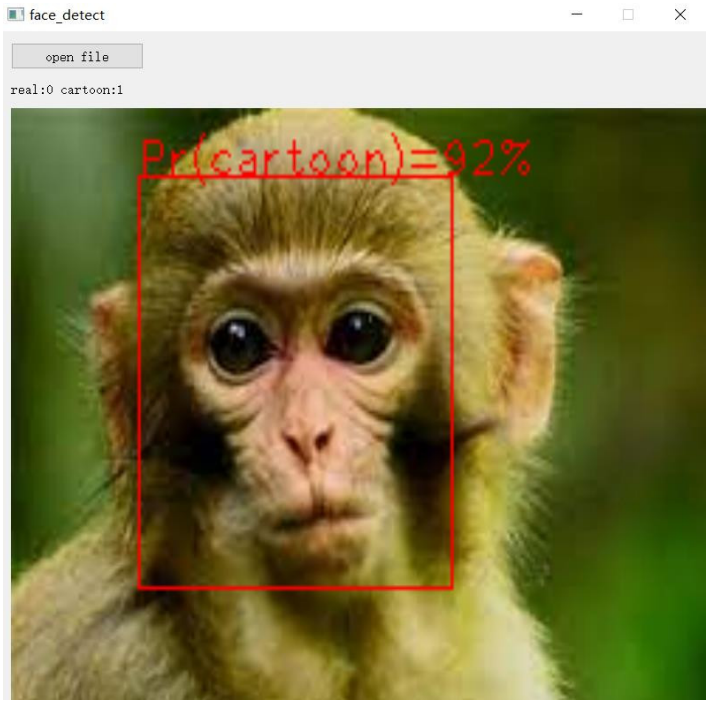


Figure 14: The generated result of utilizing the proposed method to test monkey facial image.

We summarize two key factors for the reasons why our proposed method is able to produce the above results.

1) The main reason for generating this result is caused by the use of the MTCNN model

for the face detection tasks in this project. Specifically, in the user interface, we set a judgment to check whether or not a given image contains faces (i.e., the faces include both real and cartoon faces). If there is no face in the image, the system should pop up a notification window telling the operator that the system cannot detect any faces (shown in Figure 15). In this figure, we input a landscape image to the proposed integrated model, and there is no face in this given image. After processing, the system gives a notification that can be seen in Figure 15. Therefore, if we input an image that only contains monkey faces, the output should be the same as the result in Figure 15. However, our experimental result shows that some monkey faces can be detected from the images using the MTCNN model, which should be improved in the future.

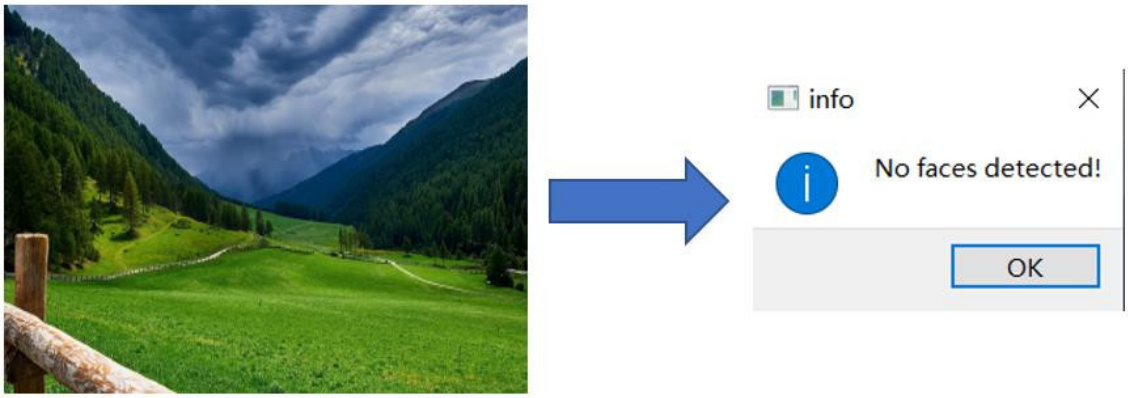


Figure 15: The generated result of utilizing the proposed method to test scenery pictures.

2) Another factor that makes our proposed method think of a monkey face as a cartoon facial image is that our collected training dataset is not large enough. If we could collect more cartoon facial images, then the above issue (shown in Figure 14) would not

happen again.

5.2: Future improvements

In this section, we will discuss several improvements that may make our proposed integrated approach better in terms of performance.

1) The first improvement is that we should collect more images for the training process. In general, training a CNN model requires a large amount of data. However, in this project, we only collected about 1000 images as our dataset. Although we use the transfer learning technique to reduce the impact of data shortage, our proposed method can still get more accurate results if we have a large amount of training data.
2) Another improvement is that we can try to train a new model without adding the pre-trained model into it. Although this improvement will be very time- consuming, the final model may be more flexible in dealing with our problems.
3) According to our experiments, some facial images of monkeys can be detected by the MTCNN model. However, this condition should not happen in face recognition systems. In order to deal with problem, we should consider the basic architecture of MTCNN model and its original training and testing datasets. Some of them should be improved and modified such that the model can obtain better results on face detection tasks.

References

[1] Adjabi, I., Ouahabi, A., Benzaoui, A., & Taleb-Ahmed, A. (2020). Past, present, and future of face recognition: A review. *Electronics*, *9*(8), 1188.

[2] Kremic, E., & Subasi, A. (2016). Performance of random forest and SVM in face recognition. *Int. Arab J. Inf. Technol.*, *13*(2), 287-293.

[3] Gonzalez, R. C., & Woods, R. E. (2008). Digital image processing: Pearson international edition.

[4] Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., Esesn, B. C., Awwal, A. A., & Asari, V. K. (2018). The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*.

[5] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[6] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, *23*(10), 1499-1503.

[7] Crumpler, W. (2020). How accurate are facial recognition systems---and why does it matter. *Center for Strategic and International Studies*, *14*.

[8] Glorot, X., & Bengio, Y. (2010, March). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (pp. 249-256). JMLR Workshop and Conference Proceedings.

[9] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision* (pp. 1026-1034).

[10] LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K. R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade* (pp. 9-48). Springer, Berlin, Heidelberg.

[11] Saxe, A. M., McClelland, J. L., & Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

[12] Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). PMLR.

[13] Taigman, Y., Yang, M., Ranzato, M. A., & Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1701-1708).

[14] Sun, Y. (2015). *Deep learning face representation by joint identification-verification*. The Chinese University of Hong Kong (Hong Kong).

[15] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815-823).

[16] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

[17] Saxena, C. (2021, March 3). Binary cross entropy/log loss for binary classification. *Analytics vidhya.* Retrieved October 30, 2021, from https://www.analyticsvidhya.com/blog/2021/03/binary-cross-entropy-log-loss-for-binary-classification/

[18] Singh, H. (2021, March 16). How images are stored in the computer? *Analytics vidhya.* Retrieved October 30, 2021, from https://www.analyticsvidhya.com/blog/2021/03/grayscale-and-rgb-format-for-storing-images/

[19] Lee, S. Y., Tama, B. A., Moon, S. J., & Lee, S. (2019). Steel surface defect diagnostics using deep convolutional neural network and class activation map. *Applied Sciences*, *9*(24), 5449.