A Mobile Application of NBA Player Face Recognition

COMP90055 Computing Project (25 Credits)
Type of Project: Software Development Project

Supervisor: Richard Sinnott
872500 Chih-Ting Su
872312 Yin-Hsiang Liu

2019 Semester 1

Table of Contents

We certify that

- this thesis does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university, and that to the best of our knowledge and belief it does not contain any material previously published or written by another person where due reference is not made in the text.
- where necessary we have received clearance for this research from the University's Ethics Committee and have submitted all required data to the Department.
- the thesis is 5673 words in length (excluding text in images, table, references, and appendices).

Acknowledgement

We want to give us special thanks to our supervisor Prof. Richard Sinnott for giving us supports and this opportunity to work on this challenging but interesting project. We learned a lot from the developing process.

Besides, we would like to thank all contributors of these open source tools, architectures and images. We won’t be able to complete this project without these materials. Thank you.

Abstract

Deep learning is one of the most popular technologies in computer science. The combination of deep learning and computer vision has brought researches in visual tasks to a new level. Deep learning allows computer vision to become more accurate that even more accurate than human visions. Convolutional neural network (CNN) is a branch of deep neural network and it is seen as the state-of-the-art architecture in computer vision, natural language processing and speech recognition. Transfer learning is a methodology that overcomes the isolated learning problem and utilized the knowledge learned from one task to solve related tasks. Transfer learning especially benefits for building complicated deep learning models by using pre-trained models.

This research report applied CNN architecture through transfer learning method to build an NBA player face recognition mobile application. There are many famous open-source CNN architectures like InceptionV3, MobileNet and VGGNet. After comparing the feasibility of mobile device and accuracy, we used VGGNet and MobileNet architectures by using Keras. To reduce the interference of the training process, all the player images that were crawled from google were captured only face section. The deep learning model identifies players based on face features of the player and the results achieve around 95 percent accuracy.

An iOS application was designed to implement our deep learning models. Users can either upload pictures from the photo library to identify NBA player or turn on the camera to carry out real-time face recognition.

Keywords: Computer Vision, Face Recognition, Deep Learning, Keras, MobileNet, VGGNet, iOS Application

1 Introduction

This report aims at building an iOS mobile application of face recognition based on deep learning models that are trained by using Keras.

1.1 Deep Learning

Deep Learning has become a popular technique these years, especially after the huge progress of hardware equipment, scientists can develop more complicated models with hardware support. Deep learning is a machine learning method using artificial neural networks, inspired by the human brain [3]. The difference between traditional machine learning and deep learning is that machine learning model extracts feature manually, but deep learning extracts feature and classify classes automatically through the model. Given a set of input data, deep learning model implements feature extraction and classification automatically, and then output the predicted label. Deep learning has been widely applied and produced extraordinary results in areas such as image processing, natural language processing and speech processing. Comparing deep learning and machine learning in the image processing area, deep learning is the-state-of-art of image processing, since it is impossible for a human to detect the huge number of features in an image with hundreds of dimensions.

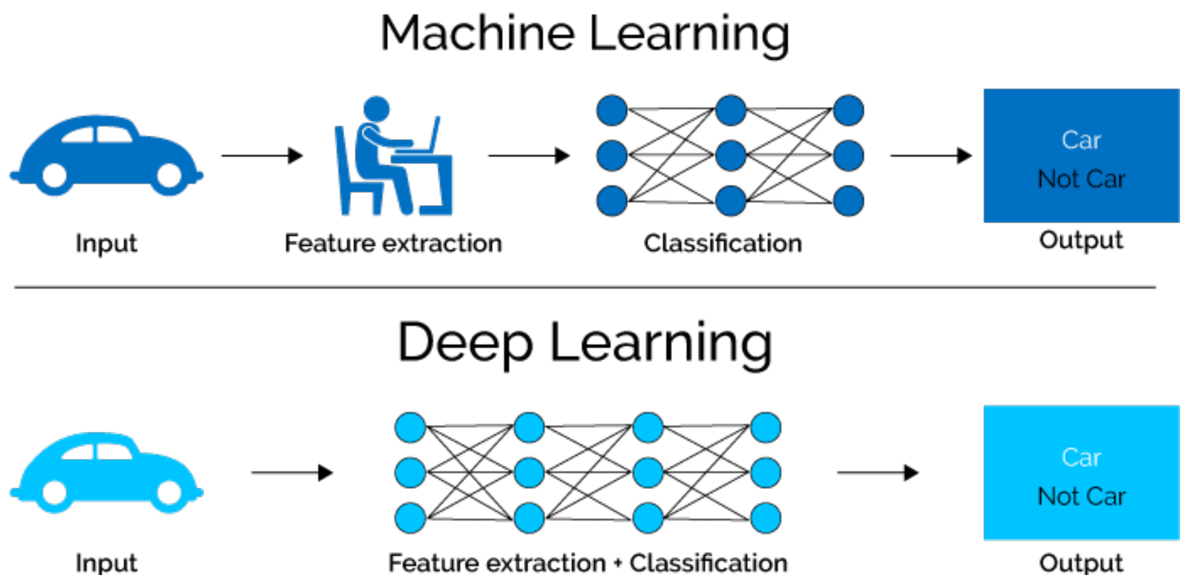


Figure 1. The difference between machine learning and deep learning [5]

1.2 Face Recognition

Face recognition identifies a person by analyzing, calculating and comparing the features of people's facial features. Face recognition is normally a step after face detection, face alignment and feature extraction. However, we didn't implement face detection in our model, face detection is done by the mobile application part because iOS provides useful API to carry out face detection.

Face Recognition technology was firstly developed in 1964 by Bisson and Wolf. The project extracted features from pictures manually. In 2012, a breakthrough deep learning model based on convolutional neural networks called AlexNet won the first prize of ImageNet LSVRC, and it reached an unprecedentedly high accuracy. Afterward, face recognition models generally based on CNN models.

1.3 Keras

Keras is an open sourced high-level neural networks API that capable of running on top of TensorFlow, CNTK and Theano [4]. It enables fast experiment through user friendliness, modularity, and extensibility. Keras allows developers to build a deep learning model with the least lines of code and the last time. The reason why developers need least lines of code with Keras is that Keras already built in various type of neural network layers, like convolutional networks and recurrent networks, developers only need to import the structure and put right parameters(weights). Also, Keras works with Python, which is simple and easy to debug.

1.4 Related Work

1.4.1 CNN and RCNN

Convolutional Neural Network (CNN), which is a branch of deep neural networks, is the most widely used technique of image recognition and image classification. As shown in figure 2, there are three main layers in CNN, which are convolutional layer, pooling layer and fully connected layer. Generally, the first layer is Convolutional Layer. Convolutional layer applies filters, which composed of kernels, to extract features. The second layer is Pooling Layer or called subsampling layer, which performs max pooling that extracts the maximum value of a certain region or average pooling that gets the average value of a specific region. The main usage of pooling layer is reducing dimensionality. Convolutional layer and pooling layer normally are combined to learn features. Developers can put several sets of convolutional layer and pooling layer to extract more features. The third layer fully connected layers are placed before the classification output. Fully connected layers are used to classify the input image into different class based on the extracted features.

Region-based Convolutional Neural Networks solves the problem of selecting a huge number of regions by using selective search to extract a manageable number of regions from the image [6]. Firstly, RCNN generates a group of about 2,000 possible regions, then extracts and saves features via pre-trained CNN models. Next, using Support Vector Machine (SVM) classifier to distinguish whether it is an object or a background. Finally, the bounding box position is corrected through a linear regression model.
However, RCNN is too heavy for a mobile application that it consumes too much memory and computing power, since every region needs to run through CNN model, so we have to run at least 2000 times CNN. Also, it cannot be implemented in real-time as it takes around 47 seconds to test an image.
In this report, we decided to use CNN model despite RCNN because the object detection task is done through the mobile application, so we don’t need the object detection feature of RCNN, CNN is lighter and more suitable for us.

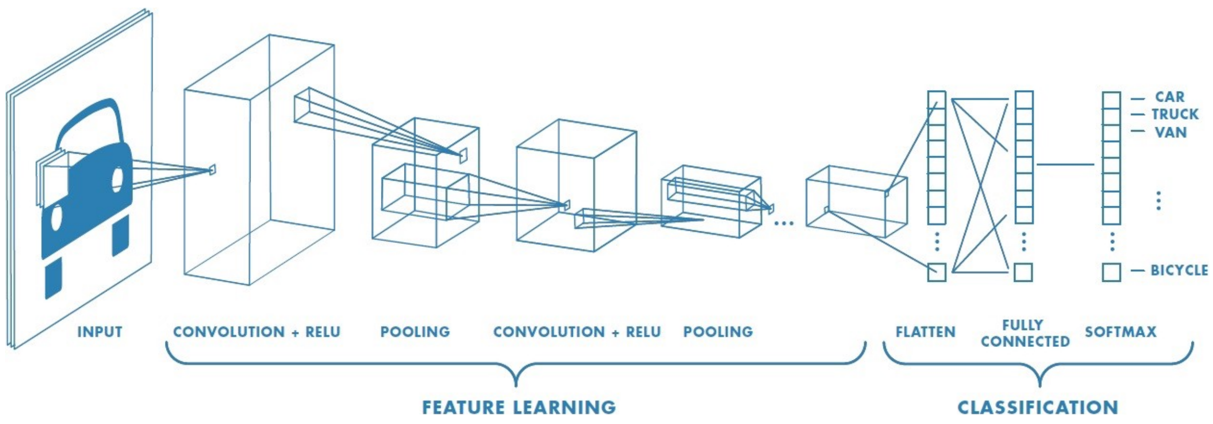


Figure 2. Structure of CNN

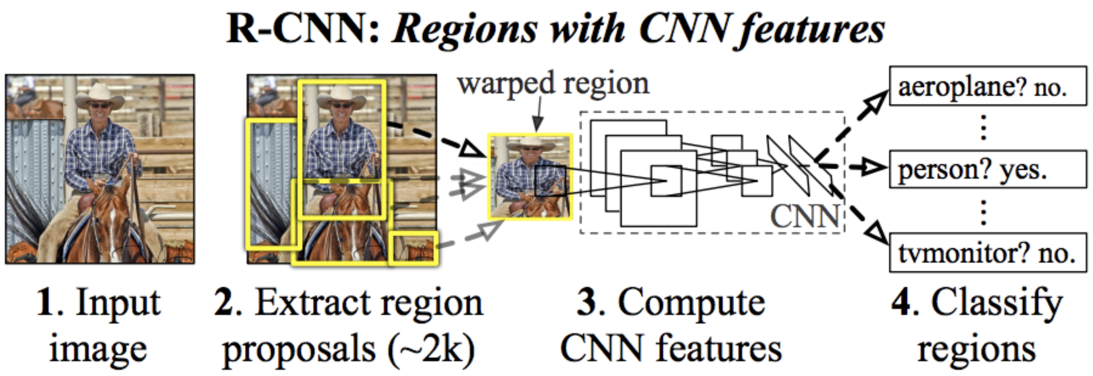


Figure 3. Structure of RCNN

1.4.2 CNN Architectures

After the success of AlexNet at ImageNet Large Scale Visual Recognition Challenge (ILSVRC) for vision analysis competition, ConvNet (CNN) architectures have become the main stream in ILSVRC, such as the very deep convolutional networks for large-scale image recognition by Visual Geometry Group and Inception architecture by GoogLeNet.

AlexNet

AlexNet was the winner of ILSVRC in 2012, afterward, ConvNet became a popular model. AlexNet contains eight learned layers: five convolutional and three fully-connected [7]. The innovative feature of AlexNet is AlexNet applies the non-linear Rectified Linear Units (ReLUs) as the activation function. Deep convolutional neural networks train several times faster with ReLUs than linear activation functions [7], therefore, ReLUs became the most popular activation function and be widely used in other ConvNet architectures.

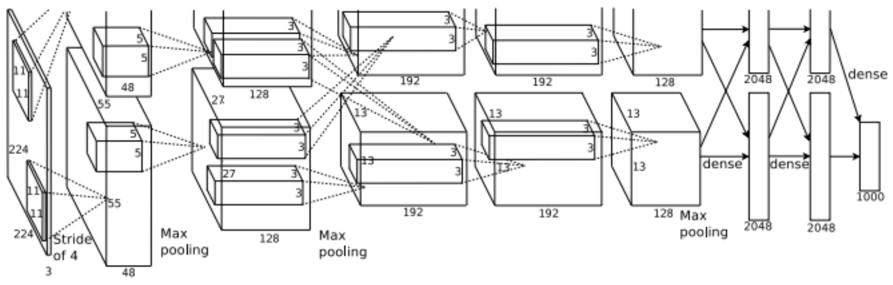


Figure 4. Structure of AlexNet

VGGNet

VGGNet is similar to AlexNet but applies more layers, which consists of 16 to 19 convolutional layers. With deeper layers, the accuracy increased, and it proved the relationship between the depth of CNN and the performance [8].

Inception (GoogLeNet)

GoogLeNet by Google was the champion of ILSVRC classification competition. GoogLeNet replaces the simple convolutional layers with a structure called "Inception" aims at broadening the width of the structure rather than the depth like VGGNet. As shown in figure 5, the structure of Inception is combining three different sizes of convolutions (1*1, 3*3, 5*5) with a pooling layer. Through the Inception structure, GoogLeNet can extract more features and details of images.

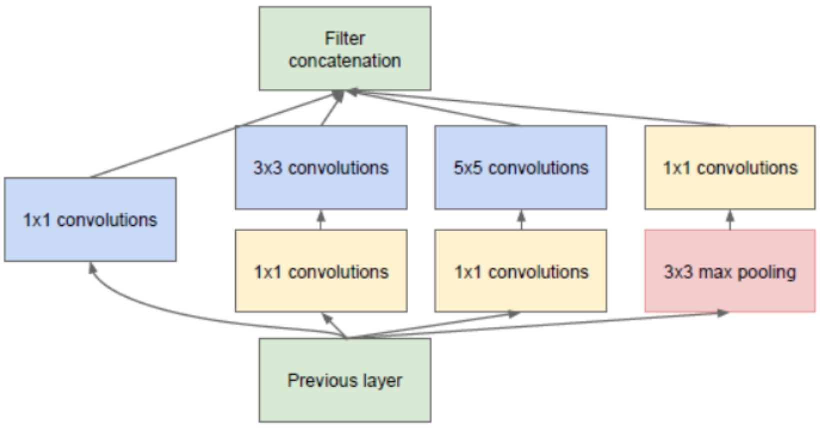


Figure 5. Inception Module with dimension reductions [9]

MobileNets

With the fast growing of deep learning, deep learning models became larger and larger, which is too heavy for mobile application or couldn't even run on the mobile devices. Therefore, Google proposed a class of efficient models called MobileNets for mobile devices. MobileNets is a lightweight deep neural network that capable of object detection, fine-grain classification and face attributes [10]. MobileNets are built primarily from depthwise separable convolutions and subsequently used in Inception models to reduce the computation in the first few layers [10]. MobileNet applies 3*3 depthwise separable convolutions which reduce 8 to 9 times computation than standard convolutions at only a slight decrease in accuracy [10]. The MobileNetV2 has seen as the state-of-the-art architecture of mobile-oriented model because it provides high accuracy and low latency.

1.4.3 Transfer Learning

Human learners are capable of transferring previous experience and knowledge between different tasks. However, traditional machine learning algorithms are normally designed isolated. Transfer Learning aims at changing this problem by developing methods to transfer knowledge learned from previous tasks and reused it as the starting point of target task to improve learning in the target task [11]. With transfer learning, fewer data and less time are needed to train a deep learning model.

Due to deep learning tasks demand a huge amount of data and computation power, transfer learning is an especially useful methodology within deep learning in computer vision. By transfer learning in computer vision, developers can use basic functions that already trained by pre-trained models, for example, object detection.

2 Dataset

One of the most crucial factors that significantly affect the performance of the trained model in deep learning is the quality and the quantity of the dataset. Therefore, we aim to acquire adequate dataset to form a solid foundation for this application. However, there is no off-the-shelf dataset for us to use, which made us acquire and build dataset on our own.

According to Basketball-Reference.com, which is a site providing professional basketball statistics, there are 530 players have a record of appearance on the NBA court. Moreover, there are over 3,000 players who ever played in the league in the past 50 years. Consider the limitation of the resource and the scope of the project; our application ends up only support to recognize 52 players. One of the reasons why we choose to support less number of players is the quality, and the quantity of the dataset, the player who is not popular or has a really short career in the league tend to have less image to collect thus decrease the accuracy of successful recognition. The other reason is the workload of gathering dataset. Labelling images and cleaning dataset is manual works, excessive classes of images would profoundly affect the process of the project. With these concerns, we have made a trade-off on the coverage of targets. The dataset contains fifty players who are the top ten players in the league at his position and with the addition of two legendary players, Kobe Bryant and Michael Jordan. Apart from the standard classes, we added a class unknown which contains random faces to give our application a way out when it cannot identify the person in the image.

2.1 Data Sources

Since there are not off-shell images dataset for NBA players, we acquire images from the web by utilizing Microsoft's Bing Image Search API with a python script to programmatically download images via queries. This approach is completed by feeding a list of players, and the API would form a query to search images on the internet than download them to the folder with the name of the player.
For the dataset of unknown class, we chose to use an existing dataset from Labelled Faces in the Wild Dataset [12], which contains 13,233 images of 5,749 people, and randomly select images to form the unknown class.

2.2 Data Pre-processing

After collecting the raw data from the web, it is necessary to keep the data align with the problem that we aim to solve. Thus, the cleaning and filtering the raw data are implemented.

A. Filtering Images

There are two main problems to solve in this stage. First one is the irrelevant images, which can be arisen because of many reasons, such as there is another person who has the same name as the player. This type of problem has a seriously detrimental effect on performance. The unrelated images would lead the model to learn the features that are useless to identify the intended target. The second problem is the duplicated images, which may cause bias when predicting. To tackle these problems, we manually identify and remove invalid images.

B. Face Alignment and Cropping Images

When building image dataset for facial recognition, proper normalization of data will benefit the algorithm to perform the task. Since our training target is a human face, the background of an image is not useful when extracting feature. To accomplish these goals, we implemented face alignment to all the photos in our dataset. The algorithm would detect the face in the picture, crop it, and give it a proper rotation. Figure 6 illustrates an example of the implementation of face alignment and cropping.

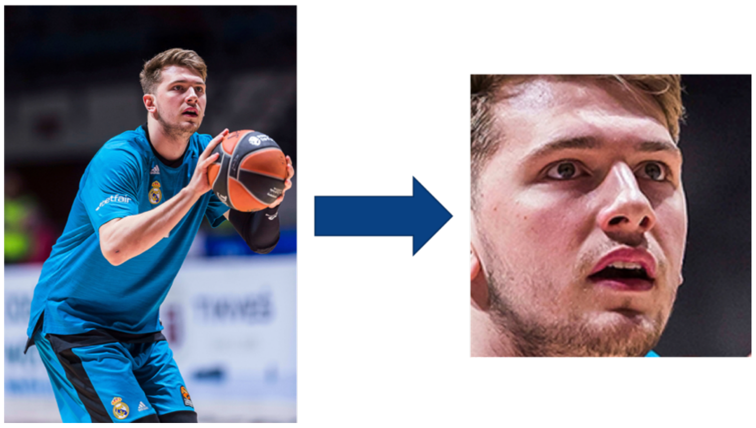


Figure 6.

C. Resize

The images downloaded from the web usually have different sizes, which will be problematic when feeding them into our network. Therefore, we must resize it to accommodate our model. The size that accepted by most image classification model is 224*224 pixels, which is the exact size of images in our dataset after adapting resizing.

2.3 Data Summary

Figure 7 represents the distribution of the number of images for each player. The total amount of images is 6,680. As shown in the graph, the distribution is not even, mostly caused by duplication and irrelevancy. This unbalanced may influence the accuracy of our model, which will be discussed later in Section 4.2.

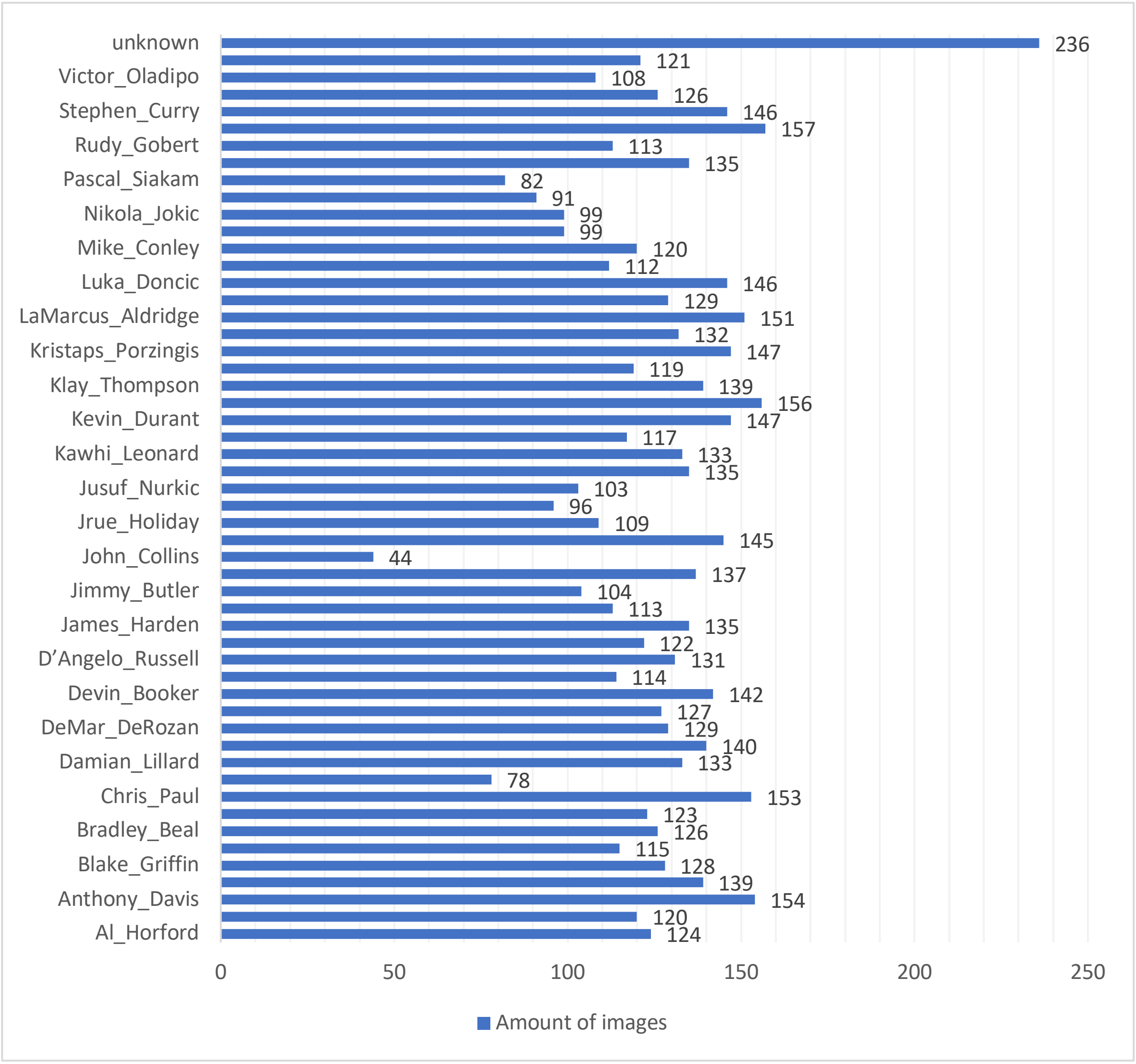
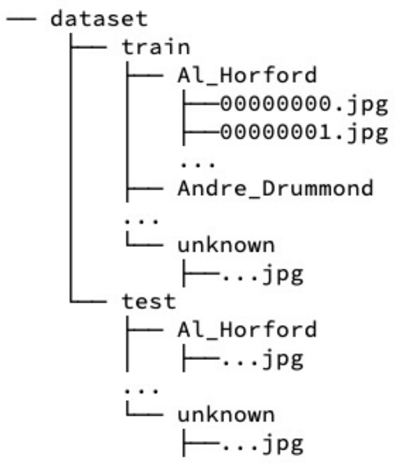


Figure 7.

The structure of the dataset is shown in Figure 8. We first divided the dataset into a training set and test set with a ratio of 8:2. The folder name represents the class of images which are in that folder.


```
── dataset
    ├── train
    │   ├── Al_Horford
    │   │   ├──00000000.jpg
    │   │   ├──00000001.jpg
    │   │   ...
    │   ├── Andre_Drummond
    │   ...
    │   └── unknown
    │       ├──...jpg
    └── test
        ├── Al_Horford
        │   ├──...jpg
        ...
        └── unknown
            ├──...jpg
```
Figure 8. The structure of the dataset

2.4 Data Augmentation

In the machine learning world, more than enough data is the key to improve the performance of the model. Thus, usually an image classification task would take thousands of images per class. Compared with the ordinary circumstance, the scale of our dataset is insufficient. Therefore, data augmentation is a meaningful way to resolve this problem. The benefit of data augmentation not only to increase the size of the dataset but also to prevent overfitting problem in deep learning and to avoid the model to extract not compelling features thus improve performance in imbalanced class problems [13].

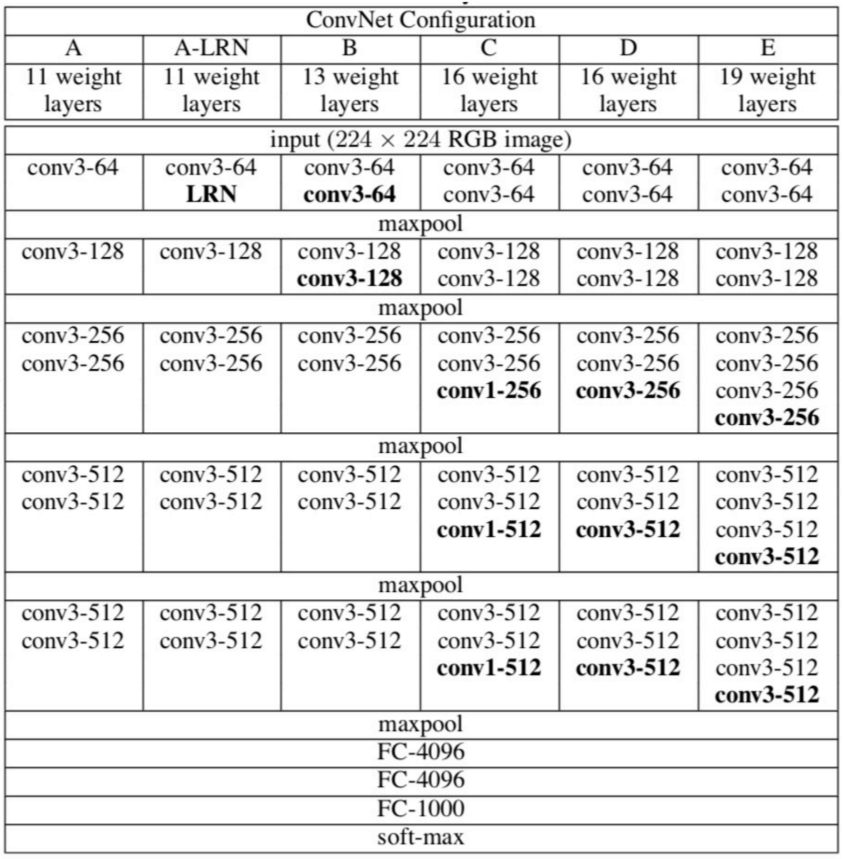
Many techniques are existing in the field; we conduct four of them, which are rotation, translation, crop, and flip to enhance the robustness of the model.

3 Training Methodology

In this project, we are leveraging two different approaches to train our model. The first one is a modified version of VGGNet, which refers to the state-of-the-art model-VGG16. The second, we utilize the pre-trained model-MobileNet-V2 with the technique of transfer learning.

3.1 VGGNet

VGGnet is a well-known model for large scale of image classification. It is invented by Visual Geometry Group from the University of Oxford in 2014 in the paper Very Deep Convolutional Networks for Large-Scale Image Recognition [12]. One of the models proposed in the paper, VGG16, achieves 92.7% top-5 test accuracy in ImageNet, which is a sizeable public image repository contain over 14 million images belonging to 1000 classes [13].
The architecture of all the ConvNet configurations present in the paper is shown in Table 1. VGG16 is constructed with 16 weight layers (13 convolutional layers and three fully connected layers).



ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64	conv3-64	conv3-64	conv3-64	conv3-64
	LRN	**conv3-64**	conv3-64	conv3-64	conv3-64
maxpool					
conv3-128	conv3-128	conv3-128	conv3-128	conv3-128	conv3-128
		conv3-128	conv3-128	conv3-128	conv3-128
maxpool					
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
conv3-256	conv3-256	conv3-256	conv3-256	conv3-256	conv3-256
			conv1-256	**conv3-256**	conv3-256
					conv3-256
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 1.

Although VGGNets perform well in terms of accuracy on the classification task, there are drawbacks of VGGnet. First, it is prolonged to train. Second, the size of the weights is over 533 megabytes. Consider the final model should be fit into a mobile device; a typical VGGNet model cannot deliver the goods. To overcome these difficulties, we decided to build our model, which has properties illustrated in Figure 9 This VGGNet like model consist of 6 convolutional layers and two fully connected layers. We shrink down the depth of the model to reduce the training time and the size of the model.

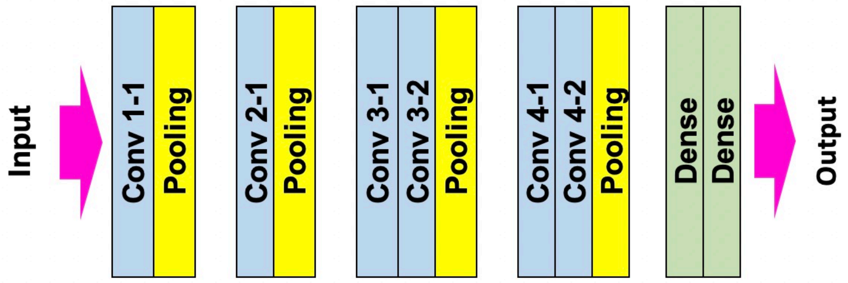


Figure 9.

The training process can be observed from Figure 10 in the training stage, we split the training dataset to training set and validation set with the ratio 9 to 1. After trained through 60 epochs, the model has achieved over 91% of validation accuracy.

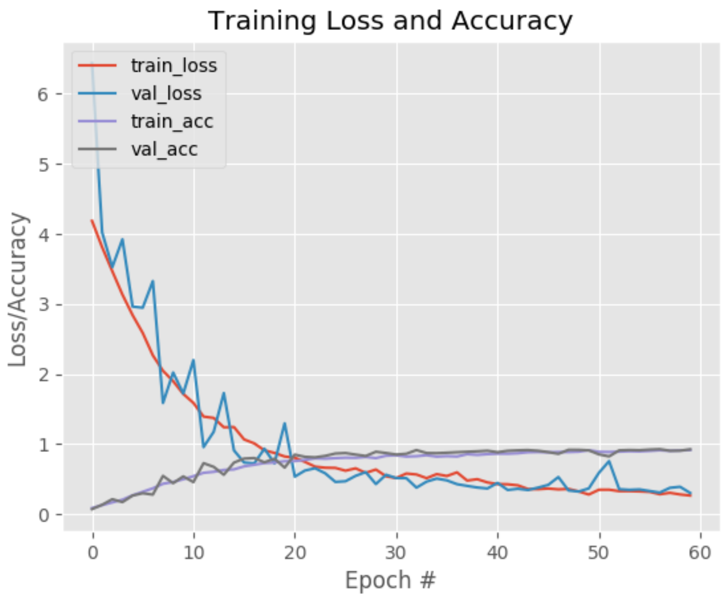


Figure 10.

3.2 Transfer Learning with MobileNet V2

3.2.1 MobileNet V2

The second experimental approach is to utilize a transfer technique with MobileNetV2, which is a computer vision neural networks inherited from MobileNetV1. This second-generation model has made significant improvements in every aspect compared with its state-of-the-art predecessor on a mobile device [15].

On account of the promising statement that Google made, we also include it as a candidate for our application. In this framework, we first import the pre-trained model with weights obtained from ImageNet and train the model with our dataset. Then apply fine-tuning to refine our model.

3.2.2 Transfer Learning and Fine Tuning

Transfer learning is a commonly seen technique to perform computer vision task. This technique lets us do not have to go through all the training process from scratch but to use the pre-trained model as a starting point in order to save immense computing power and massive training time. In [16], the experiment result shows that transfer learning often helps when performing a similar task. From [17], the result proves that transfer learning has the potential to accomplish better performance with limited dataset compared with the model which without leveraging the technique.

For our model, we chose the MobileNetV2 as a pre-trained model with weights from ImageNet, then re-train it with our dataset with custom classifier layers. After several iterations, we freeze the weights of first few layers since them capturing universal features which are not specific to our problem, and then train the rest layers with smaller learning rate to avoid distortion on pre-trained weights.

The ratio of the training set and validation set is 9 to 1; the model achieves validation accuracy of 93.4%.

4 Outcome Analysis

In this project, we experimented two different frameworks to produce our classification model. The first is VGGNet with custom depth of layers, and the second is MobileNetV2 with transfer learning. The performance of the two models will be presented and compared with tables and histogram in the following sections.

4.1 Accuracy analysis

The performance test was conducted with the test set, which stands along from the dataset we use in the training stage, i.e., the models have never seen these images before. We examine the model by feeding test set images into the model then it will make a prediction. We compare the prediction with the label of the test instance to judge if it is a correct prediction. The testing results were recorded and compiled to form the diagrams for each model. Table 2 and 3 for VGGNet and MobileNetV2, respectively. The comparison of the two models can also be found in Table 4.

Name	Test case	Correct	Incorrect	Accuracy
Al_Horford	25	23	2	92%
Andre_Drummond	24	23	1	96%
Anthony_Davis	31	31	0	100%
Ben_Simmons	28	26	2	93%
Blake_Griffin	25	25	0	100%
Bojan_Bogdanovic	23	21	2	91%
Bradley_Beal	25	25	0	100%
Brandon_Ingram	24	23	1	96%
Chris_Paul	29	28	1	97%
Clint_Capela	15	15	0	100%
Damian_Lillard	26	25	1	96%
Danilo_Gallinari	28	28	0	100%
DeMar_DeRozan	25	25	0	100%
DeMarcus_Cousins	25	25	0	100%
Devin_Booker	28	28	0	100%
Donovan_Mitchell	23	23	0	100%
D’Angelo_Russell	26	25	1	96%
Giannis_Antetokounmpo	24	24	0	100%
James_Harden	27	27	0	100%
Jayson_Tatum	22	22	0	100%
Jimmy_Butler	21	21	0	100%
Joel_Embiid	27	27	0	100%
John_Collins	8	8	0	100%
John_Wall	29	29	0	100%
Jrue_Holiday	22	21	1	95%

Julius_Randle	19	19	0	100%
Jusuf_Nurkic	19	18	1	95%
Karl-Anthony_Towns	27	27	0	100%
Kawhi_Leonard	25	25	0	100%
Kemba_Walker	23	22	1	96%
Kevin_Durant	29	28	1	97%
Kevin_Love	31	30	1	97%
Klay_Thompson	27	27	0	100%
Kobe_Bryant	24	24	0	100%
Kristaps_Porzingis	29	24	5	83%
Kyrie_Irving	26	24	2	92%
LaMarcus_Aldridge	29	29	0	100%
LeBron_James	26	25	1	96%
Luka_Doncic	29	25	4	86%
Michael_Jordan	22	22	0	100%
Mike_Conley	24	24	0	100%
Montrezl_Harrell	20	19	1	95%
Nikola_Jokic	20	20	0	100%
Nikola_Vucevic	18	18	0	100%
Pascal_Siakam	16	16	0	100%
Paul_George	26	26	0	100%
Rudy_Gobert	22	19	3	86%
Russell_Westbrook	31	31	0	100%
Stephen_Curry	29	29	0	100%
Tobias_Harris	24	21	3	88%
Victor_Oladipo	21	21	0	100%
Zach_LaVine	24	24	0	100%
unknown	46	44	2	96%
Total	1316	1279	37	97%

Table 2. Accuracy of the VGGNet model of each player

Name	Test case	Correct	Incorrect	Accuracy
Al_Horford	25	24	1	96%
Andre_Drummond	24	21	3	88%
Anthony_Davis	31	30	1	97%
Ben_Simmons	28	26	2	93%
Blake_Griffin	25	24	1	96%
Bojan_Bogdanovic	23	23	0	100%
Bradley_Beal	25	25	0	100%
Brandon_Ingram	24	21	3	88%
Chris_Paul	29	27	2	93%
Clint_Capela	15	14	1	93%
Damian_Lillard	26	26	0	100%
Danilo_Gallinari	28	27	1	96%
DeMar_DeRozan	25	23	2	92%
DeMarcus_Cousins	25	24	1	96%
Devin_Booker	28	28	0	100%
Donovan_Mitchell	23	21	2	91%
D'Angelo_Russell	26	26	0	100%
Giannis_Antetokounmpo	24	22	2	92%
James_Harden	27	27	0	100%
Jayson_Tatum	22	20	2	91%
Jimmy_Butler	21	19	2	90%

Joel_Embiid	27	27	0	100%
John_Collins	8	5	3	63%
John_Wall	29	27	2	93%
Jrue_Holiday	22	19	3	86%
Julius_Randle	19	17	2	89%
Jusuf_Nurkic	19	19	0	100%
Karl-Anthony_Towns	27	26	1	96%
Kawhi_Leonard	25	25	0	100%
Kemba_Walker	23	21	2	91%
Kevin_Durant	29	29	0	100%
Kevin_Love	31	30	1	97%
Klay_Thompson	27	23	4	85%
Kobe_Bryant	24	22	2	92%
Kristaps_Porzingis	29	28	1	97%
Kyrie_Irving	26	24	2	92%
LaMarcus_Aldridge	29	29	0	100%
LeBron_James	26	23	3	88%
Luka_Doncic	29	26	3	90%
Michael_Jordan	22	21	1	95%
Mike_Conley	24	24	0	100%
Montrezl_Harrell	20	20	0	100%
Nikola_Jokic	20	18	2	90%
Nikola_Vucevic	18	17	1	94%
Pascal_Siakam	16	14	2	88%
Paul_George	26	23	3	88%
Rudy_Gobert	22	21	1	95%
Russell_Westbrook	31	27	4	87%
Stephen_Curry	29	29	0	100%
Tobias_Harris	24	24	0	100%
Victor_Oladipo	21	21	0	100%
Zach_LaVine	24	22	2	92%
unknown	46	46	0	100%
Total	1316	1245	71	95%

Table 3. Accuracy of the MobileNetV2 model of each player

Model	Test case	Correct	Incorrect	Accuracy
VGGNet	1,316	1,279	37	97%
MobileNetV2	1,316	1,245	71	95%

Table 4. Comparison of two models

4.2 Accuracy and Uneven dataset

How much data would be enough to train the model in terms of prediction performance of the machine learning model is a hard question which seemed not have reached a consensus in the field. In [14], the author had done experiments to evaluate the result of the models which trained with different size of the dataset. The research result shows the models constructed using bigger datasets outperformed the model constructed with the smallest dataset and achieves better accuracy. This result infers that using sufficient data can lead to better performance. Since our dataset possesses the property of uneven distribution, we investigate the results to discover whether we construct our model with the adequate size of a dataset. The graphs are shown below:

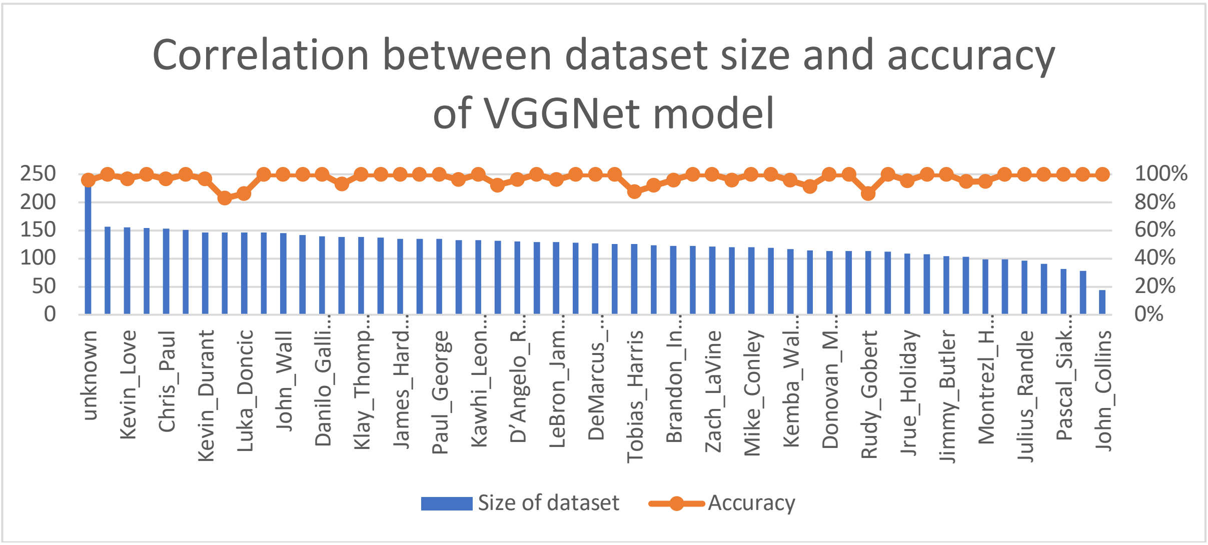


Figure 11.

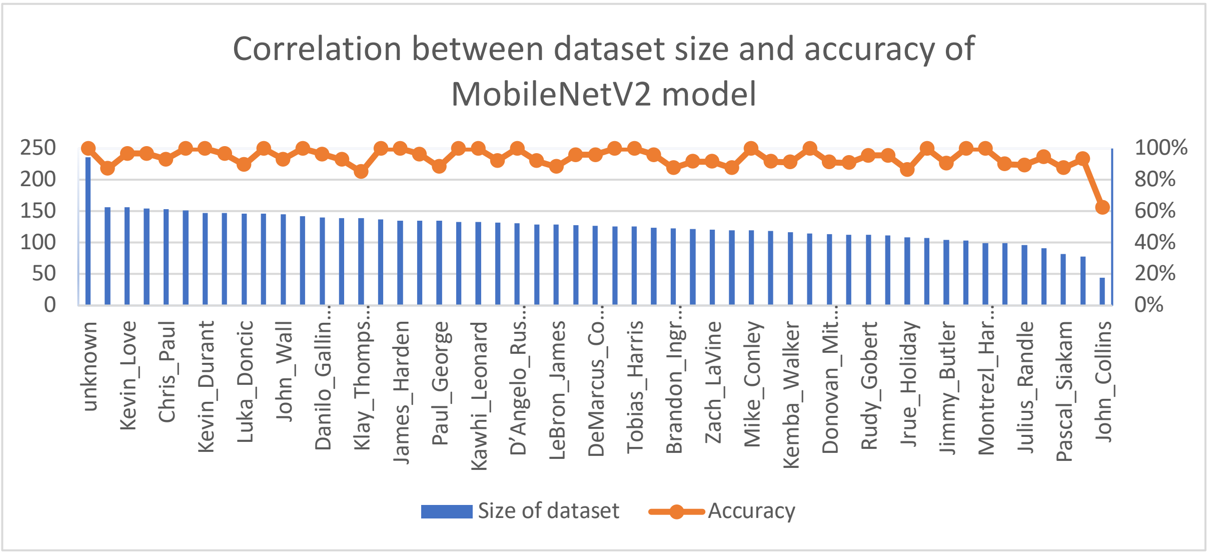


Figure 12.

In Figure 11 and 12, we can observe that the two models express differently in the test set, the accuracy of each class from VGGNet seems not affects by the size of dataset, while some of the accuracies of classed from MobileNetV2 significantly dropped (Entire model: 95%, John_Collins: 63%) when the dataset size of the class is significantly smaller than others (Average size: 126, John_Collins: 44).

This phenomenon may infer that for the VGGNet model, the size of the dataset is enough while as it is not the case for MobileNetV2 model. To achieve better performance, acquire more data is required.

5 Implementation on Mobile Environment

This section introduces how we designed the mobile application, and how to implement our deep learning model on the mobile device.

5.1 Platform Choice

There are many choices of the mobile application environment, and there are some reasons why we choose iOS over Android. Firstly, iOS has no fragmentation. iOS has much fewer models of devices, which means there are much fewer screen sizes and fewer platforms to support. Secondly, it is easy to simulate the application by simply connecting the iPhone with the computer.

Also, Swift provides many fantastic and convenient packages, for example, CoreML. CoreML is a machine learning framework that used on Apple products for performing fast prediction and easy to convert third party machine learning model like Keras and Caffe into CoreML format. With CoreML, developers only need a few lines of code to implement the machine learning model into the mobile application. Furthermore, CoreML supports Vision for image analysis and real-time predictions, which meet our need for this project. Also, Core ML is optimized for on-device performance, which minimizes its memory footprint and power consumption [1].

5.2 Keras on iOS

CoreML provides the service of converting Keras trained model into CoreML format file. After converting into CoreML file, we can simply implement our deep learning model in iOS application in a few lines of code. Also, we can simply replace different models by changing the variable name in the code. Since CoreML doesn’t support TensorFlow, Keras is more suitable for our project.

5.3 User Interface

For implementing in the mobile application, user interface design is critical. The goal of this application is providing users an excellent interface and a satisfying operating experience. To let users get started easily, the design of the user interface is simplified to make users understand everything at a glance. As shown in figure 13, there are two buttons: Upload and Camera on the welcome page. In this application, users can either upload images from their photo library or turn on cameras to carry out real-time face recognition.

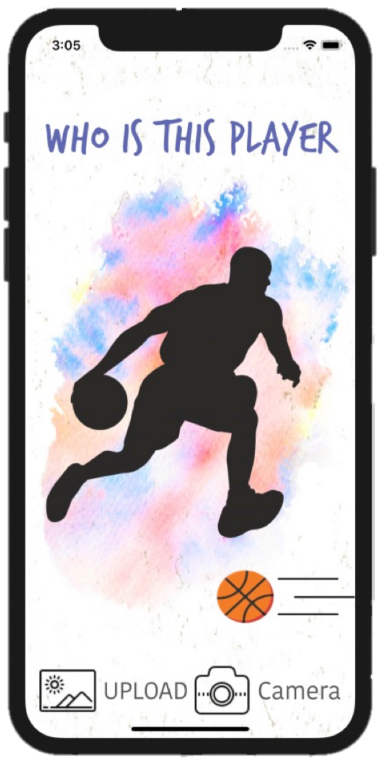


Figure 13.

5.3.1 Photo Library Upload

After the user clicked the Upload button, users can choose images from their photo library and upload images to the system. The system will identify the face in the entire picture, crop the face and then send the face section to the deep learning model to recognize the result. If users upload an image without a human face or the player is not in our dataset, the system will show "couldn't find any face" and "unknown".

Since the APP is "face" recognition, face cropping is fairly important. If we input the whole picture into the model, there will be too much interference in the picture that leads to a biased result. To perform face cropping, we add an SDK called FaceCropper by using CocoaPods. FaceCropper detects the face automatically and crops the face part.

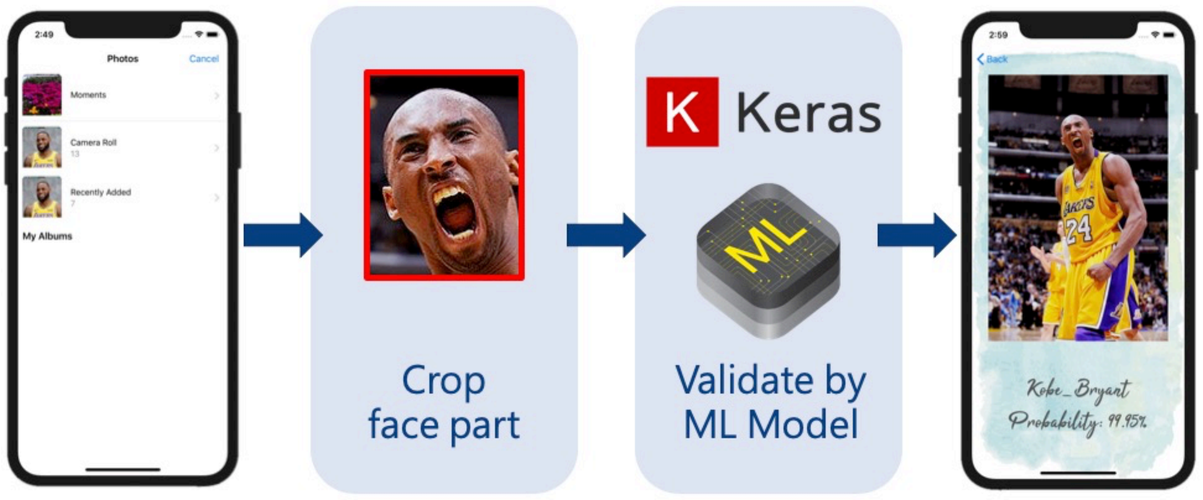


Figure 14.

5.3.2 Camera Real Time

The second way to do face recognition is by using the camera to do real-time face recognition. As the camera moves, the system continuously recognizes the face on the screen, also displays the predict player name and probability. The structure and function were inspired by Rosebrock [2]. In the real-time section, we applied AVFoundation, which is a framework provides service for working with time-based audiovisual media and capturing session, then pass it to the deep learning model. In the real-time face recognition, we didn't put face cropping function, because we assume that people normally will focus their camera on the face part.

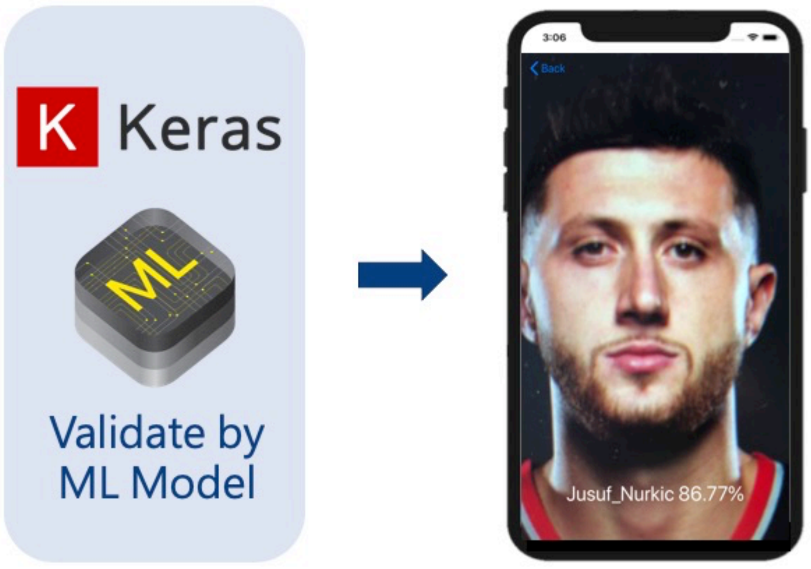


Figure 15.

6 Conclusions

This research report introduces an implementation of an iOS application of NBA player face recognition to help people who are new to NBA games recognize current top 50 players. Firstly, we introduced the techniques we used in this report and a list of architectures in related works. Based on the result of previous works and the feasibility of implementing the deep learning model on mobile devices, we chose to train two models based on VGGNet and MobileNet. Secondly, 6680 images of around 50 players were crawled from Google and automatically labelled, pre-processed and cropped the face section. Thirdly, we trained and compared two deep learning models based on CNN architectures: VGGNet and MobileNet by using transfer learning method. The results of these two models perform remarkably with 97 percent accuracy and 95 percent accuracy. We only implement 8 layers in the VGGNet model, and it outperforms MobileNet. Next, we implement these two models in our iOS application. The trained weights of VGGNet takes 53.8 MB and MobileNet only takes 14.5MB. Both are lightweight enough for implementing on mobile devices. We also introduce our user interface and features of the iOS application. Finally, we proposed that there are some future works can be improved or added.

In summary, a deep learning model of face recognition is capable of implementing on mobile devices with excellent user experience and remarkable accuracy at the same time. In this application, both VGGNet and MobileNet have satisfying results.

7 Future Works

Although we have reached an excellent accuracy of our model, some parts can be improved or be added.

1. At the moment, the application can only recognize one person and one face at a time. In the future, we want to add the function that makes it able to recognize multiple players no matter from the picture or real-time camera.
2. There are only around 50 players in our dataset, these are the 50 best players today. However, the player that users don't know might not be that famous, so it would be better if we can increase the number of players.
3. Although these players are the 50-best player, some of the players get fewer media coverage, therefore, it is difficult to get a sufficient amount of data. It is an issue that we need to figure out a solution.
4. Sports lovers may not only like to watch one specific sport. Therefore, the application will be more functional if add a function of recognizing other sports players, like football players or baseball players in the future.

Appendix

Demo Video

https://youtu.be/jj4eyGNRow4

Source Code

https://github.com/liuyinhsiang/Face-Recognition-iOS-app-on-NBA-players

Reference

[1] Core ML. (n.d.). Retrieved from https://developer.apple.com/documentation/coreml

[2] Adrian Rosebrock. Running Keras models on iOS with CoreML. (2018, April 23). Retrieved from https://www.pyimagesearch.com/2018/04/23/running-keras-models-on-ios-with-coreml/

[3] Gulli, A., & Pal, S. (2017). *Deep Learning with Keras*. Birmingham: Packt Publishing.

[4] Keras: The Python Deep Learning library. (n.d.). Retrieved from https://keras.io/

[5] Tommy Huang. What is AI, Machine Learning and Deep Learning. https://medium.com/@chih.sheng.huang821/什麼是人工智慧-機器學習和深度學習-587e6a0dc72a

[6] Gandhi, R. (2018, July 10). R-CNN, Fast R-CNN, Faster R-CNN, YOLO-Object Detection Algorithms. Retrieved from https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[7] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84-90. doi:10.1145/3065386

[8] Karen Simonyan, Andrew Zisserman. VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION. ICLR 2015

[9] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2015). Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). doi:10.1109/cvpr.2015.7298594

[10] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam: MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. CoRR abs/1704.04861 (2017)

[11] Torrey, L., & Shavlik, J.W. (2009). Chapter 11 Transfer Learning.

[12] Jessica Li. Retrieved from https://www.kaggle.com/jessicali9530/lfw-dataset

[13] Sebastien C. Wong, Adam Gatt, Victor Stamatescu, Mark D. McDonnell: Understanding Data Augmentation for Classification:When to Warp?DICTA2016:1-6

[14] A. R. Ajiboye, R. Abdullah-Arshah, H. Qin, H. Isah-Kebbe. Evaluating the Effect of Dataset Size on Predictive Model Using Supervised Learning Technique. International Journal of Software Engineering & Computer Sciences (IJSECS) 2015

[15] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. doi:10.1109/cvpr.2018.00474
[16] Rosenstein, M. T., Marx, Z., Kaelbling, L. P., & Dietterich, T. G. (n.d.). To Transfer or Not to Transfer
[17] Huang, Z., Pan, Z., & Lei, B. (2017). Transfer Learning with Deep Convolutional Neural Network for SAR Target Classification with Limited Labeled Data. Remote Sensing,9(9), 907. doi:10.3390/rs9090907