東海大學

資訊工程研究所

碩士論文

指導教授: 楊朝棟博士

遷移式學習應用於正顎手術前後對稱性評估

Using Transfer Learning for Facial Symmetry Assessment Before and After Orthognathic Surgery

研究生: 鄭鈞澤

中華民國一零八年六月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 鄭 鈞 澤 所提之論文

遷移式學習應用於正顎手術前後對稱性評估

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

召 集 人 陳世言 簽章

委 員 [illegible]

[illegible]

[illegible]

指 導 教 授 楊朝棟 簽章

中華民國 108 年 6 月 10 日

摘　要

遷移式學習是目前影像辨識相關研究中對於小數據集訓練中非常重要的方法，我們將遷移式學習應用於醫療領域，透過卷積神經網路將針對正顎手術前後進行面部對稱性的評分，評分標準的部分我們經由長庚醫院的 8 位醫師問卷調查，將醫師的評分做為參考標準，但由於面部對稱性評分沒有一定的標準，因此我們將評分做平均並去除 1 至 3 個標準差之外的值，讓評分更加一致性，透過處理電腦斷層掃描 3D 影像將其轉換為等高線圖，此等高線圖保有 3D 的特徵，由於我們資料量過小所以需要使用資料預處理，我們透過不同的資料預處理方法與資料擴增，將原始資料擴增為 100 倍，並透過實驗找出最符合我們目標任務的方法，並於本文中使用了四種預訓練模型，本文實驗中比較了四種模型的優劣，而實驗中使用的神經網路為將預訓練模型導入後，增加全連結層以及分類層，並生成隨機變形資料訓練時放入模型，使用此方法以及加入 Dropout 神經網路層來防止模型過度擬合，由實驗的結果我們最終選擇了 Xception 模型以及 Constant 的資料擴增方法達到高達 90% 的準確率，透過模型預測時產生的信心值，給出模型所預測的評分。本文完成了使用遷移式學習進行訓練，達到針對正顎手術前後的立體顏面影像萃取之等高線圖片進行對稱度的評分，並比較了不同的預訓練模型以及不同預處理方法的優劣，進而找出真實準確率最高的方法。

關鍵字: 遷移式學習，卷積神經網路，深度學習，面部對稱性，資料預處理

Abstract

Assessing facial symmetry is crucial for orthodontists. An accurate appraisement of face symmetry is notable for the development of dentofacial orthopedics diagnosis. To facilitate a successful treatment, an understanding of personal characteristic in the perception of face symmetry become the critical factor. However, there is no standard of facial symmetry score. It depends on the orthodontist's expertise in face symmetry judgments. Therefore, it is tough to ensure accuracy. To support the physicians for improving the medical treatments, in this paper, we propose a Convolutional Neural Networks(CNN) with transfer learning method for facial symmetry assessment based on three-dimensional features. In this case, we train the new model through Transfer learning to score facial symmetry. In this work, we convert the computed tomography (CT) scan of a 3D image into a contour map which retains the characteristics of 3D. To find the best result, we use different data pre-processing method and data amplification method. The original data is amplified by 100 times. In our experiment, we compare the quality of the four models, and the neural network architecture used in the experiment is to import the pre-training model. Also, we increase the fully-connected layer and the classification layer. To prevent the model from overfitting, we put the random deformation data during training and Dropout. From the experimental results, we chose the Xception model and the Constant data amplification method to reach up to the accuracy rate of 90%. The score predicted by the model is given by the confidence value predicted by the model.

Keywords: Transfer Learning, CNN, Deep Learning, Facial symmetry, Data Pre-process

致謝詞

完成本篇論文需要感謝很多人，最重要的是要感謝我的指導教授楊朝棟博士，在碩士這兩年的點點滴滴，從起初對人工智慧相關的研究感到興趣，到現在使用人工智慧技術完成我的論文，中間遇上了各種難題，努力的研究與學習讓我對人工智慧影像辨識相關的領域有了一定程度的專業，非常謝謝老師在各方面的指導以及各方面的教誨，讓我不僅僅是在學業方面有所成長，也在待人處事有了很大的進步。謝謝長庚醫院顱顏醫學研究中心林秀霞博士的合作，在研究上提供的意見以及指導，對於問題皆詳盡地回答，讓我能夠完成本篇論文。

謝謝口試委員陳牧言教授、呂芳懌教授、黃國展教授、欉振坤教授百忙之中抽空參加我的論文口試，每個教授的寶貴意見都能夠讓我的碩士論文更加完善。也謝謝高效能的學長姐、學弟們的指教與幫忙。也謝謝實驗同屆的同學們，非常幸運能夠有你們一起學習成長，一起奮鬥打拼學習新的技術，最重要的是在遇到困難時互相幫忙及討論，讓看事情的角度有了新的面向，進而解決難題，讓我們兩年的研究所時光能夠如此的精采。

最後要謝謝我的家人一直以來的支持，讓我在學習的過程中無後顧之憂地往前，無條件的支持我念完碩士，在各方面的關心以及我遇到低潮時的意見，給我繼續往前的動力，謝謝一路支持我的人，給我建議還有幫助讓我能夠順利的完成論文。

東海大學資訊工程學系 高效能計算實驗室 鄭鈞澤 一零八年六月

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the era of artificial intelligence, deep learning is applied in various fields. The growth of deep learning has made it possible for many tasks that could not be discerned by machines in the past. The breakthrough of deep learning in image recognition and each algorithm derivatives enable all fields to develop related applications. For example in the field of image processing, deep learning used to help the philosopher, artist to study about the face attractiveness [1].

The standard for facial symmetry has always been very subjective. There is no standard equation for facial symmetry. Chung et.al said that a lot of previous studies have employed composite faces to study the effects of symmetry and averageness of the face on attractiveness [2–6]. Based on Wen-Chung et.al this research will use Transfer Learning to learn and predict about facial symmetry.

For the training of small datasets, it is necessary to use the technology of Transfer Learning. Through the powerful feature extraction function of the pre-trained model, add new classifier that meet requirement. Also, the enhancement and preprocessing of the data set are an indispensable part .The goal of this paper is to use 3D CT image files, pre-processing of images combined with CNN deep learning to train facial symmetry features, and construct a facial symmetry classifier to achieve automatic prediction of facial symmetry score.

1. The 3D CT scan images of 71 subjects were converted to 2D contour maps. Contour map of 20 lines.

2. The contour map data is used as an image Pre-process, including rotation, scaling, and clipping for data amplification.

3. The pre-trained model uses Transfer Learning to train a new facial symmetry feature model.

4. Predicting facial symmetry scores for new data using new trained model

1.1 Motivation

The rise of deep learning image recognition in the field of artificial intelligence allows machines to learn like humans. In order to achieve our goal, the use of machine learning models can not learn deep features, so this thesis chose the method of deep learning. In the field of image recognition, the facial symmetry evaluation before and after the Orthognathic surgery is done by the doctor, which is takes times. The evaluation results are subjective for doctor. Therefore, this thesis hope that through the use of CT 3D images, which contain parts of soft and hard tissues, in the objective judgment, the facial features can be trained through deep learning. Let the machine have facial symmetry standard to predict the facial score before and after the operation, it can save the time. In the research of facial symmetry, most of them use the traditional method. This thesis use 3D images for image pre-processing and convert them into facial contour maps to preserve the features of 3D images. This thesis use MATLAB as a tool to do preliminary image pre-processing.

At the same time as the development of image recognition depth learning, Transfer learning has gradually emerged. Due to the small amount of data, even if the data is amplified to the maximum extent without affecting the characteristics, the amount of data is still far from re-establishing large depth Model, so this thesis choose to use the image recognition Transfer learning method to train. The

powerful feature extraction ability of different pre-training modules allows us to achieve the goal this thesis need in the case of an extremely small amount of data. This thesis will compare 4 different models to find the best method. Use the found method to train a facial symmetry scoring model to help doctors complete facial symmetry assessment tasks.

1.2 Thesis Goal and Contributions

In this work, this thesis propose a facial symmetry scoring system using Transfer Learning, which is an open source architecture that uses MATLAB to perform preliminary image preprocessing of CT 3D raw image input, and completes the data amplification by Keras. The second step of image preprocessing, using CNN pre-trained model architecture and parameters to adjust the training of the new model, identification analysis to make the face symmetry score prediction.

Training Data Set this thesis have images of 20 contours of 71 subjects. Since the data set is very small, this thesis try to integrate the design and adjustment of the top-level architecture of the pre-training model through the open source architecture. Different ways of amplifying data without unduly affecting overall image integrity, different parameters to adjust and test different optimizers, Learning rate and experiment of layers to find solutions that have large differences and very small data through the method of Transfer Learning.

1.3 Thesis Organization

The structure of this paper is as follows. Chapter 2 introduces the main background and related work. Chapter 3 outlines the system design and its implementation in our framework and shows the functionality of each component in the system. Chapter 4 details the experiments and discussions. Finally, in Chapter 5, this thesis provide conclusions and future work for this paper.

Chapter 2

Background Review and Related Work

In this section, this thesis review some background knowledges for later use of system design and implementation.

2.1 Background Review

2.1.1 Deep Learning with Python

Python [7] is a widely used, literally translated, general-purpose programming language. Python is an extensible programming language that provides a lot of APIs and tools. It is very easy for programmers to design and write extension modules using C, C++ and other programming languages. Python itself can be easily and quickly integrated into other language programs. In today's era of AI development, many frameworks support the Python language to design such as TensorFlow, Caffe, MXNet, etc. Many programmers use Python as the AI development language.

2.1.2 TensorFlow

TensorFlow [8] is an open source software library that is currently used in various machine learning and deep learning developments. It was developed by Google. This article uses TensorFlow to support Keras' backend and uses GPU version of TensorFlow to accelerate the overall deep learning. Through the combination of Tensorboard to observe the overall training situation and to modify and adjust the model, in addition to the reasons for this work, it also supports its various types of kits, and many software libraries can be integrated to make development faster and more convenient. Figure 2.1 shows architecture of TensorFlow

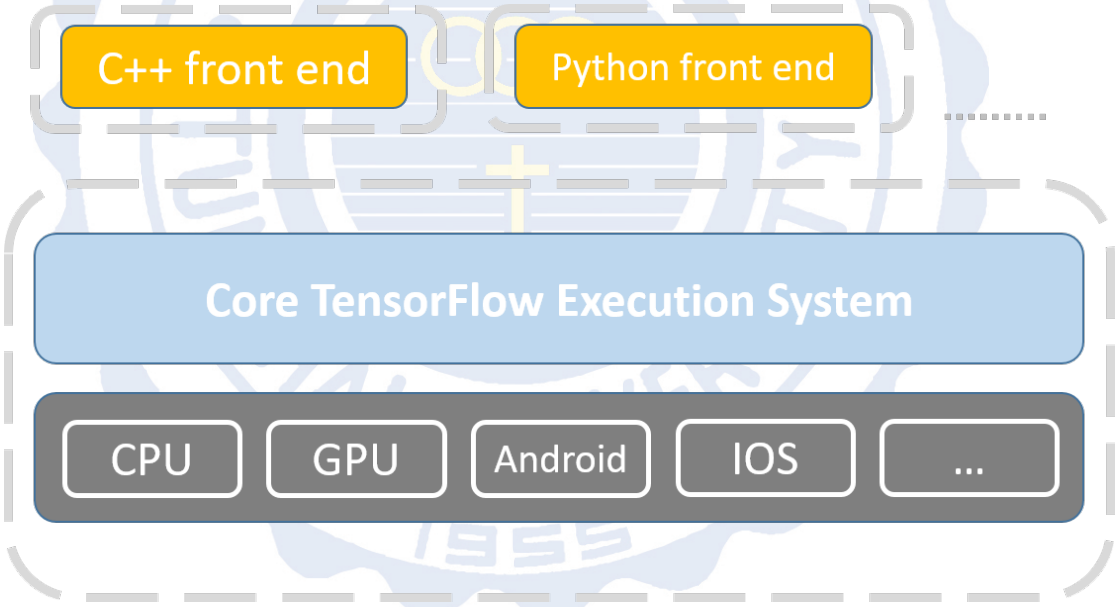


FIGURE 2.1: TensorFlow architecture

2.1.3 Keras

Keras [9] is an open source code, with TensorFlow or Theano as the back end. It is a high-level deep learning library developed in Python. This work uses Keras as the overall transfer learning model for the development environment design. Through the Keras programmer, you can pay more attention to the development and adjustment of the model. The model is very fast and intuitive to build and has considerable scalability. In addition, Keras has very high performance.

2.1.4 Facial Symmetry Data Set

The data of this work is provided by Chang Gung Memorial Hospital with CT scan samples of patients and normal persons. There are 71 samples in total. And 8 professional doctors were asked to give facial symmetry scores for these samples. This thesis use raw data processing to convert raw 3D data into contour maps. The scoring process is based on the exclusion of values that are too different in the score. Average the scores of 8 doctors. Let each sample have only one major score. This approach can make the original subjective facial symmetry score relatively objective. The detailed process of processing will be explained in Chapter 4.

For machine learning and deep learning with insufficient data, data preprocessing takes up a very important position. For different target tasks, there are many factors that lead to the raw data that need to be preprocessed to highlight features and filter noise. Too much noise can cause the model to learn the wrong features. Due to the insufficient amount of data, this work requires special pre-processing of the original data to make up for the serious shortage of data. In the case of a very large amount of data, deep learning can automatically learn the features in the image through the training of the neural layer. Traditional machine learning requires the machine to learn through artificial feature engineering. The common data preprocessing method is Binarization. HOG, zoom, rotate, etc.

This work uses the Image Data Generator in Keras and OpenCV to amplify and adjust the original data, and the original data is amplified by 100 times by rotating, scaling, and shifting. This work uses the method of rotation scaling to amplify the data set, because these two methods do not affect the original image. In deep learning, if the amount of data is too small, Problems such as over-fitting and other problems lead to overall training failure, so in this work need to augment the data set.

2.1.5 Convolutional Neural Network

Convolutional Neural Network [10], abbreviated as CNN, has outstanding performance for image processing and speech recognition. It usually consists of one or more convolution layers combined with a pooled layer and a fully connected layer. Compared with other kinds of neural networks, there are relatively few parameters to be considered, and the convolutional layer usually adds a Rectified Linear Units layer (ReLU layer), which can enhance the nonlinear characteristics of the neural network. It does not change the convolutional layer. In addition, training the neural network requires a loss function to calculate the difference between the actual value and the predicted result, to adjust the model to move in the right direction.

Figure 2.2 shows the operation flow of CNN. Through multiple convolutional layers plus pooling layer superposition through the linear rectification unit as the excitation function, the model will learn the low-order to high-order features of the graph, and finally add the fully connected layer and the classifier to classify the graph.

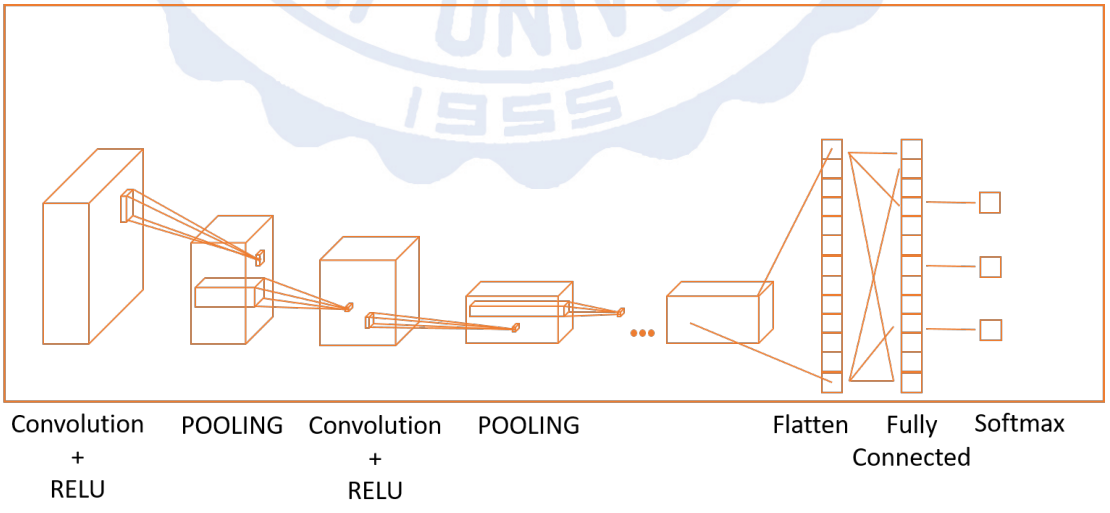


FIGURE 2.2: How CNN works

2.1.6 Transfer Learning

Transfer Learning [11] uses a pre-trained model to retrain new target tasks, just like standing on the shoulders of giants.Mainly used in new target datasets is too small, so it is impossible to retrain large depth model, which can be divided into two types:

- Train the main model (pre-trained model) yourself, use this model to do Transfer Learning to create new target models for new goals, such as training facial features as the main model (large data), and using Transfer Learning to train face recognition classification. The model becomes a new target model (small data), and the target model is built on the main model that has learned what is the face feature to train and adjust.
- Using pre-trained models trained by others, this thesis usually use this method, using the structure of the pre-trained model (big data) to train and adjust for our new goals (small data), currently there are many open source Efficient depth-pre-trained models are available, usually based on the ImageNet dataset (million images, one thousand classification), which have a minimum accuracy of over 95%.

2.2 Related Works

In the training of CNN-related models, the main reference to the application of Transfer Learning and the improvement of accuracy, as well as other CNN training related papers, The main references are as follows:

Lumini et al. [12] presented the Deep Learning and transfer learning features for plankton classification papers. They compared the advantages and disadvantages of many different pre-trained models applied to plankton classification, and the effects of different kinds of data processing methods on training. They also show how to combine different CNNs to improve performance.

Wang et al. [13] used dual-channel CNN combined with LSTM architecture for recognition for cognitive passive radar. In addition, they proposed a parameter transfer method to solve the problem of transferring parameters across various sampling frequencies, and verified the effectiveness of the method.

SandipKute et al. [14] used Transfer Learning in forensic applications, face recognition of some facial organs, they use the ears, nose, lips to find relevance and recognition, these three parts will not change due to posture or expression, They proposed a CBFR method that uses KNN as a classifier through the capture and normalization of CNN features.

Vogado et al. [15] used Transfer Learning combined with Support Vector Machine(SVM) classifier for the diagnosis of leukemia, they lack large database, so use Transfer Learning to extract features, and test the advantages and disadvantages of SVM, K Nearest Neighbor(KNN), Multilayer perceptron(MLP), random forest classifier, and finally choose SVM. The experiment also compares the quality of different CNN architectures and achieves 99% accuracy.

Ciocca et al. [16] published a paper on the retrieval and classification of food images, which used the Transfer Learning method and compared different CNN architectures ,ResNet-50 was found to be the best for food classification.

This thesis also studied different CNN methods. Liu et al. [17] proposed the architecture of Multi-view multi-scale CNNs for the classification of CT images.They propose a new convolutional neural network-based nodular classification method that can handle four classical methods of solid nodule types (ie, well- Cirmscribed, vascularized, juxta-pleuraland pleural-tail). Their method also enables competitive classification rates on ground-glass optical (GGO) nodules and non-nodules.

YuzhuJi et al. [18] proposed a multi-scale attention CNN method for the detection of Salient objects by introducing spatial and channel direction attention layers into the multi-scale encoder-decoder framework. Note that the CNN layer can align the context information between the different scales of the feature map and the final prediction of the saliency map.

Since medical-related treatments often have data imbalance problems, this thesis also refer to Li et al. [19] to propose a Deep variance network, an improved CNN framework for unbalanced data sets, and enhanced CNN generalization capabilities by subspaces. Integrating with the Bayesian network into the CNN framework, a new Deep Variance Network (DVN) is proposed, which can transfer the joint feature distribution from one object's complete training data set to other objects in an iterative manner. Complete training data set.

For solving the problem of overfitting, it has always been a major problem in the training of CNN models. Xu et al. [20] proposed a simplification of the full-link layer to solve the method of over-simulation. The author mentioned in the article that in CNN A large number of parameters allow it to learn complex features, but they may tend to hinder generalization by overfitting training data. Although many of the previously proposed regularization method overfitting is still a problem for training powerful CNN, they propose that SparseConnect mitigates overfitting by sparsely connecting to FCL, and experiments in three benchmark datasets MNIST, CIFAR10 and ImageNet.

The following are papers related to data processing. For small data volume image data collection, effective data amplification is very important. Han et al. [21] use CNN combined with Transfer Learning to propose a new image classification method. Discuss the adjustment of parameters in Transfer Learning and the amplification methods and comparison of effective image data sets. Franc et al. [22] proposed a paper on Learning CNNs from weakly annotated facial images, which automatically annotates age, gender, and identity, and links annotations to the correct face. They also built a new facial image database.

Chapter 3

System Design and Implementation

3.1 System Architecture

This thesis use Python language development, use Windows as the development environment, use Anaconda on windows to build Python 3.6 environment, first convert the original 3D CT scan image into 2D contour map through MATLAB, then rotate through Keras, Displacement, scaling to amplify data, use OpenCV to do pre-processing of data, then develop framework. This thesis build CUDA10 [23] and Cudnn7.3 [24] set up TensorFlow GPU framework as the bottom layer, top layer uses Keras combined with Matplotlib and Numpy suite as development tool training model and do Predicting. As shown in Figure 3.1

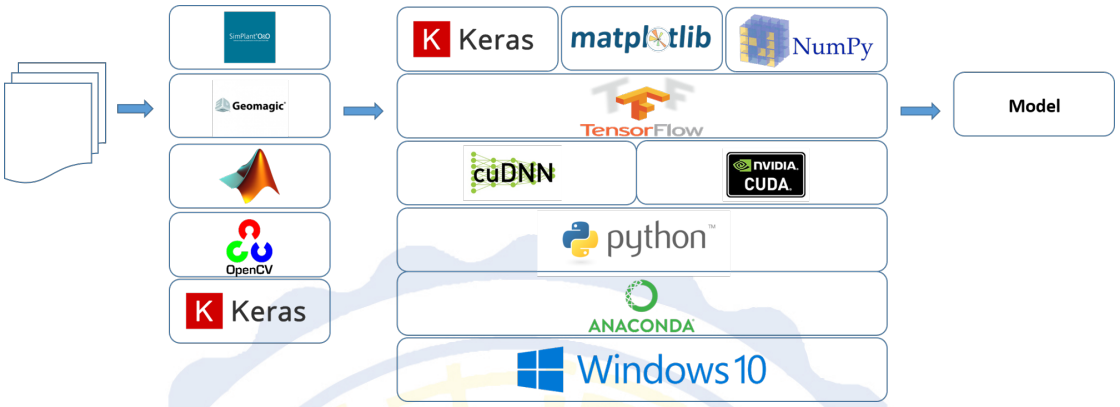


FIGURE 3.1: System analysis process

3.2 Data Pre-processing

3.2.1 Data 3D to 2D

First, obtained the medical computed tomography image of 3D without any processing. this thesis used MATLAB to do the data preprocessing of the first step, and converted the 3D file into a 2D contour map. Retained the 3D features through the color. As shown in Figure 3.2.

3.2.2 Data tailoring and feature enhancement

By using OpenCV to cut our picture as a midline, the right half of the image is horizontally flipped and superimposed to the left image as shown in Figure 3.3. Because of the lack of data, if the amount of data is too small, It must Image pre-processing allows Transfer learning to learn better features. Through the pre-processing of data, it can let the machine learn the characteristics of facial symmetry effectively.

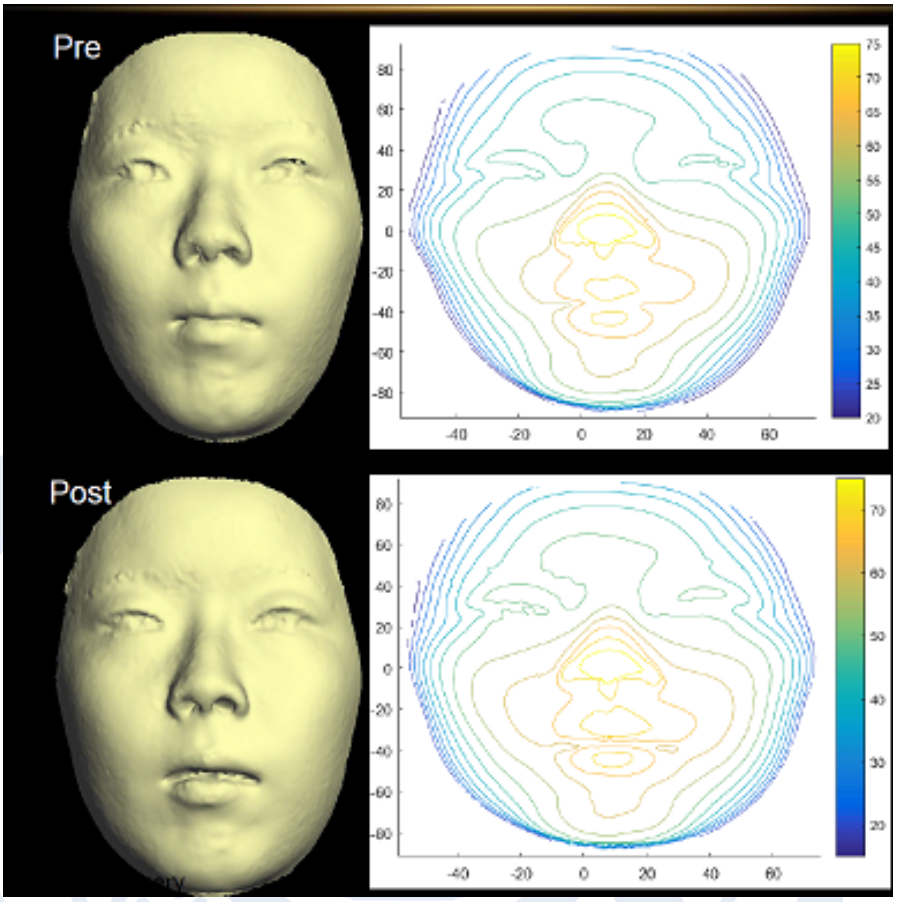


FIGURE 3.2: Convert to contour map

3.2.3 Data amplification

In order to cope with the small amount of data, this thesis use Keras data generators. To augment our dataset, it will scale, rotate, shift, etc. the image without affecting the image features, and fill the blanks generated after processing into black. The original data was a facial CT scan of 71 subjects, resulting in 20 contour maps. This thesis use this process to amplify the data by 100 times.As shown in Figure 3.4

3.3 CNN Training

At present, the research results of artificial intelligence have confirmed that CNN has better feature extraction efficiency than traditional methods. It is a very common method for image recognition to use CNN through deep learning. There

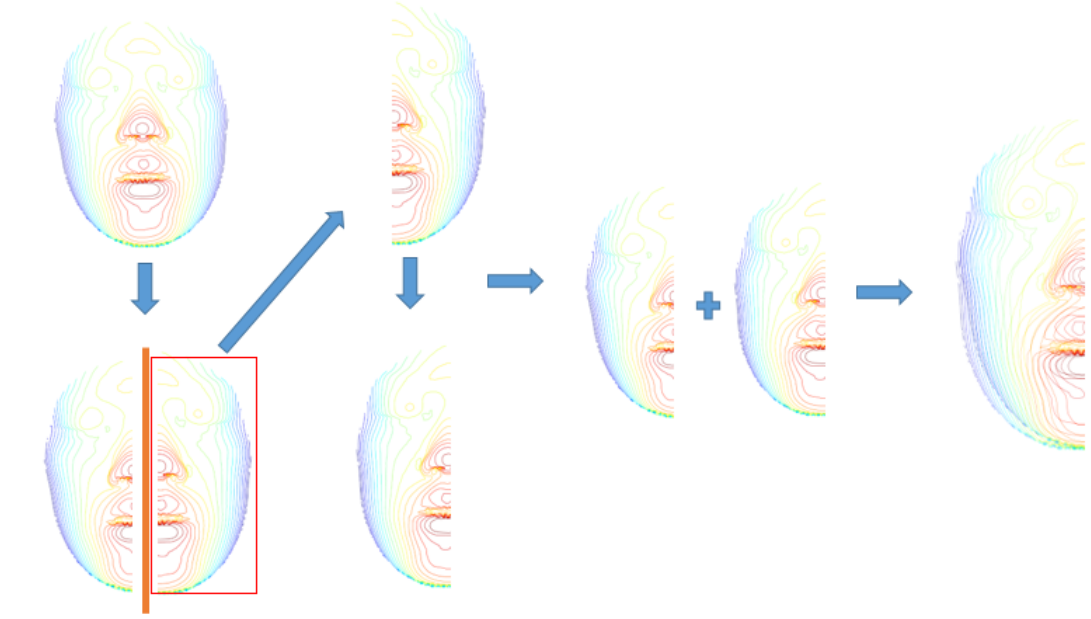
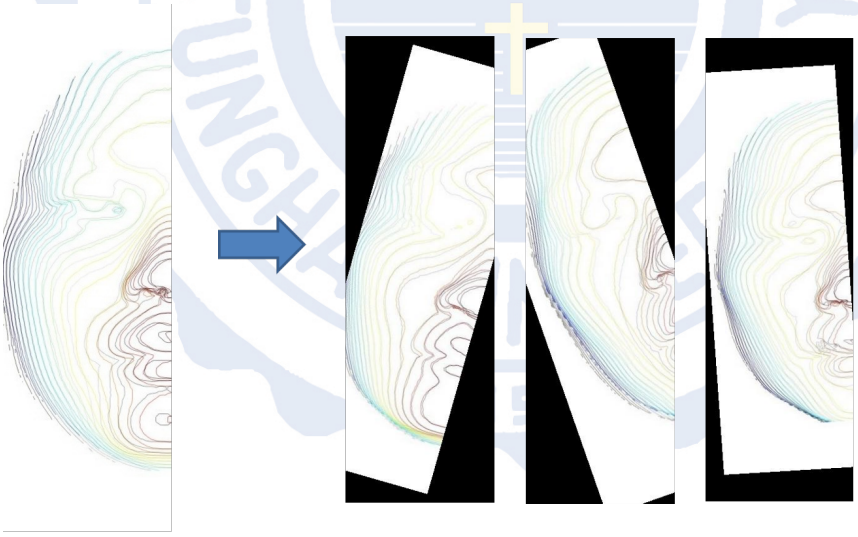


FIGURE 3.3: Feature processing



45

FIGURE 3.4: Data amplification

are two common methods for image analysis using CNN. In the first way, the deep CNN architecture training is designed through large datasets, and the model with powerful feature extraction performance is trained. At the end of the model, the task classifier is added to complete the effective classification task. The second is to conduct Transfer Learning through a pre-trained neural network, also known as a pre-trained model. In the smaller target tasks of the dataset, the powerful features extracted from the large datasets are extracted, and the new dataset is fine-tuned

to achieve the same goal. In this work, our goal is to use the contour maps that retain 3D features for facial symmetry scores, as shown in Figure 3.5,but since our dataset size does not allow us to pre-train large deep learning CNN architectures, this thesis Use the open source pre-trained training model to do the Transfer Learning approach to accomplish our mission.Below have listed some basic neural layer introductions of CNN neural network .

1. Convolution Layer : There are many convolutional units in the convolutional layer. The most important function is to extract various features from the input data. The deeper convolutional layer can capture more complex features.

2. Activation Layer : This layer is to add various different educational funtions that meet the training objectives. The main purpose is to introduce nonlinearity. The most common Activation funtion is ReLU.

3. Pooling Layer : This layer is a very important part of the convolutional neural network. The common ones are the Max pooling layer and the Average pooling layer. The function is to reduce the input data space according to the size set by the pooling layer, reduce the parameters and reduce the calculation amount. The Max pooling layer finds the maximum value in the selection range and retains it, while the Average pooling layer averages the selected area. Different pooling layers are selected according to different training objectives.

4. Fully Connected : For the last few layers in the convolutional neural network structure, the previously learned features are mapped to subsequent layers. Map to the final classification layer. This layer is also the layer with the most parameters in the neural network.

5. Classification Layer : For the final classification, the common function of the multi-classification task is the Softmax function, which is also the classification function used in this work, and the two classifications are commonly Sigmoid functions.

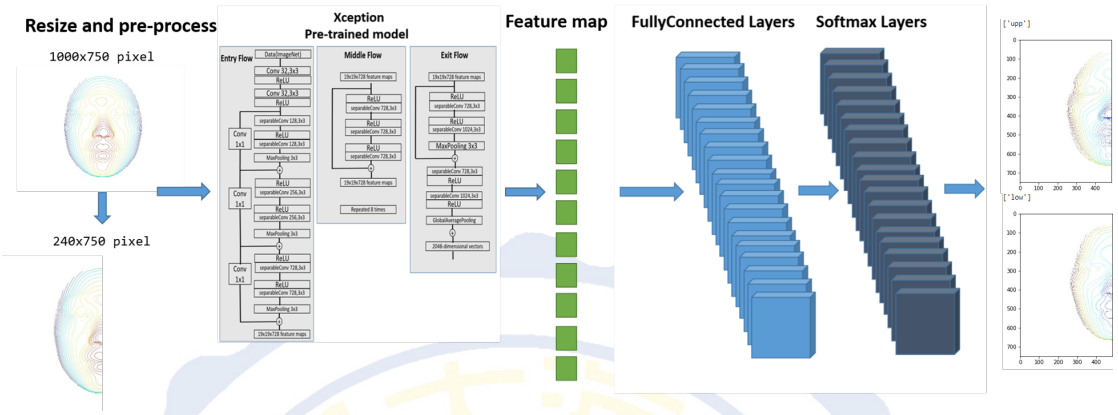


FIGURE 3.5: CNN training process

3.4 Transfer Learning with Pre-trained Model

The reason this work choose to use Transfer Learning is that the dataset is insufficient, must use Transfer Learning. In this work will compare a variety of different pre-trained models, such as VGG16 [25], VGG19 [25], ResNet50 [26], Xception [27].This work uses these four models, because these four models are very classic pre-training models. The power of the model is often used by various studies.This work will choose the model that the best result for us. The goal is Accurately give the closest rating to the doctor, and Transfer Learning has a variety of ways to adjust the model. This work will also compare the Fine-tune part of the model, the Fine-tune overall model, and the complete Fine-tune only on the top layer plus the fully connected layer and classifier. The strengths and weaknesses, through a small amount of information this work through Transfer Learning to achieve our goals.The table3.1 below shows the detailed data of each model this work will test and the accuracy of its performance on ImageNet.Below will introduce the four models this work use.Size: The highest occupancy of the memory.Top-1 acc indicates that the probability is only predicted once and correctly.Top-5 acc indicates that it is a good chance to predict the correctness of five times as long as one guess is right.

1. VGGNet: by Karen Simonyan and Andrew Zisserman. The name is derived from the Visual Geometry Group at the University of Oxford. Therefore, the model is named VGG, and the model inherits the ideas of AlexNet [28]

and builds more layers. The original version was presented in 2014 and the current version was introduced in 2015 for the sixth edition. This model won the second place in the 2014 ILSVRC Image Classification Competition. There are two versions, VGG16 and VGG19. The two models are hidden layers of 16 and 19 layers.Model architecture is shown in Figure 3.6,D and E are VGG16 and VGG19.

2. ResNet : proposed by Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, this model won the championship in the 2015 ILSVRC image classification competition, they proposed a new neural network layer called Residual. In the neural network training, in order to increase the depth to learn more complex features, but when the number of layers is increasing, the deep neural network often has a gradient disappearing problem leading to training failure. The new layer they propose skips a specific number of layers to prevent the gradient from disappearing. After this method is proposed, there are many continuation of ResNet and improved models appear.Model architecture is shown in Figure 3.7.The red box in the figure is the architecture of ResNet50, which has more layers than ResNet18 and ResNet34. The deeper architecture can learn more complex features.

3. Xception: Presented by François Chollet and the lead author of the Keras development framework. The Xception model is a variation of the Inception [29] model, and the origin of the name is the extreme version of the Inception model. This model has a special separable convolutional layer that replaces the original model structure of the Inception model. The computational efficiency of this model is very good, and the parameter size is almost as good as the Inception model.Model architecture is shown in Figure 3.8

Table 3.1: Pre-trained model detail

Model	Size	Top1 acc	Top5 acc	Parameters	Depth
VGG16	528MB	0.715	0.901	138,357,544	23
VGG19	549MB	0.727	0.910	143,667,240	26
ResNet50	99MB	0.759	0.929	25,636,712	168
Xception	88MB	0.790	0.945	22,910,480	126

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 x 224 RGB image)					
Conv3-64	Conv3-64 **LRN**	Conv3-64 **Conv3-64**	Conv3-64 Conv3-64	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 **Conv1-256**	Conv3-256 Conv3-256 **Conv3-256**	Conv3-256 Conv3-256 Conv3-256 **Conv3-256**
maxpool					
Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512 **Conv1-512**	Conv3-512 Conv3-512 **Conv3-512**	Conv3-512 Conv3-512 Conv3-512 **Conv3-512**
maxpool					
Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512	Conv3-512 Conv3-512 **Conv1-512**	Conv3-512 Conv3-512 **Conv3-512**	Conv3-512 Conv3-512 Conv3-512 **Conv3-512**
Maxpool					
FC-4096					
FC-4096					
FC-1000					
Soft-max					

FIGURE 3.6: VGGNet architecture

Layer name	Output size	18-layer	34-layer	50-layer
Conv1	112x112	7x7, 64, stride 2		
Conv2_x	56x56	3x3 max pool, stride 2		
		$\begin{bmatrix} 3x3, & 64 \\ 3x3 & 64 \end{bmatrix}$ x2	$\begin{bmatrix} 3x3, & 64 \\ 3x3 & 64 \end{bmatrix}$ x3	$\begin{bmatrix} 1x1, & 64 \\ 3x3, & 64 \\ 1x1, & 256 \end{bmatrix} x3$
Conv3_x	28x28	$\begin{bmatrix} 3x3, & 128 \\ 3x3 & 128 \end{bmatrix}$ x2	$\begin{bmatrix} 3x3, & 128 \\ 3x3 & 128 \end{bmatrix}$ x4	$\begin{bmatrix} 1x1, & 128 \\ 3x3, & 128 \\ 1x1, & 512 \end{bmatrix} x4$
Conv4_x	14x14	$\begin{bmatrix} 3x3, & 256 \\ 3x3 & 256 \end{bmatrix}$ x2	$\begin{bmatrix} 3x3, & 256 \\ 3x3 & 256 \end{bmatrix}$ x6	$\begin{bmatrix} 1x1, & 256 \\ 3x3, & 256 \\ 1x1, & 1024 \end{bmatrix} x6$
Conv5_x	7x7	$\begin{bmatrix} 3x3, & 512 \\ 3x3 & 512 \end{bmatrix}$ x2	$\begin{bmatrix} 3x3, & 512 \\ 3x3 & 512 \end{bmatrix}$ x3	$\begin{bmatrix} 1x1, & 512 \\ 3x3, & 512 \\ 1x1, & 2048 \end{bmatrix} x3$
	1x1	Average pool, 1000-d fc, softmax		
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9

FIGURE 3.7: ResNet architecture

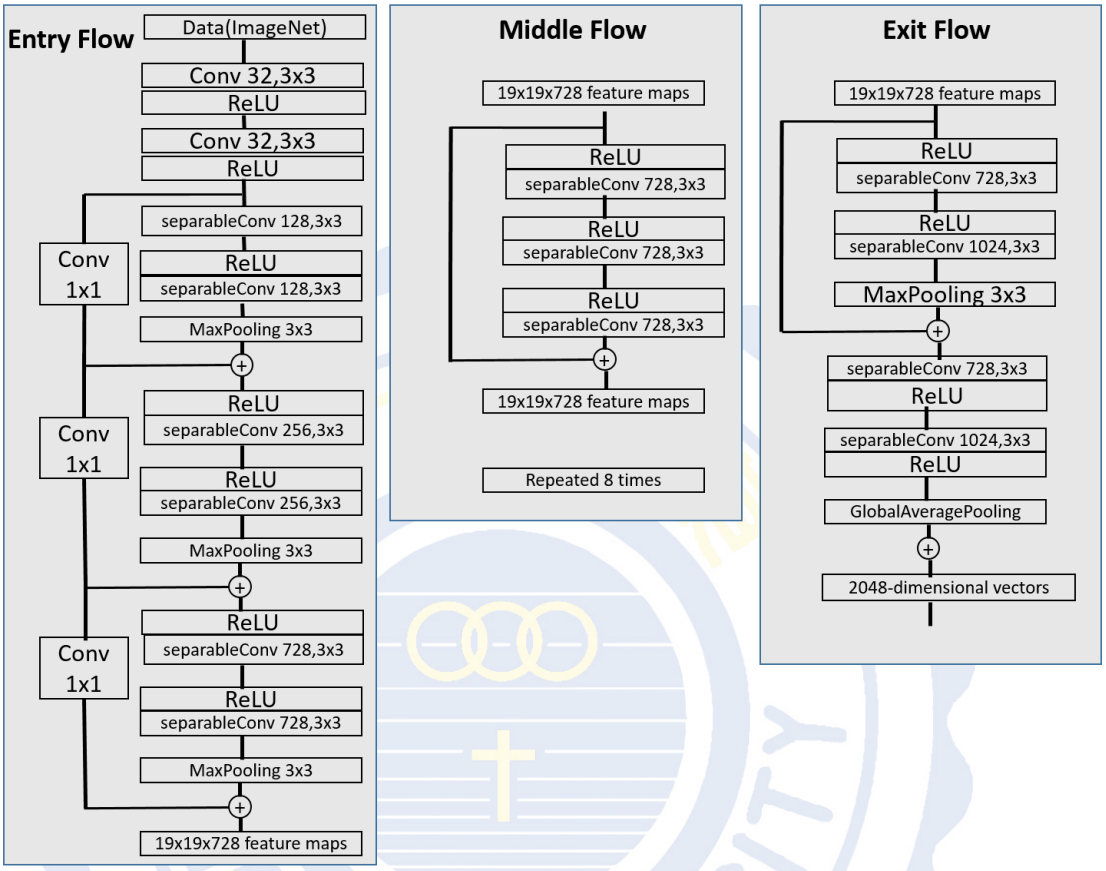


FIGURE 3.8: Xception architecture

3.5 Training Detail

The important equations and algorithms use in the model training, the Activation Function is the ReLU, ReLU linear rectification unit, which pushes the values into the first quadrant, which can effectively solve the problem of gradient disappearance, such as the equation 3.1 [30]:

$$f(x) = max(0, x) \tag{3.1}$$

The activation function of our last layer is Softmax, which is mainly used in the multi-classification target task to map the multi-neuron output between 0 and 1. This equation will help the neural network to judge and find the most likely classification. As the equation 3.2 [31]:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1...K \tag{3.2}$$

Our Loss Function uses Categorical crossentropy, a loss function, and the loss function is the residual of the actual value and the predicted value. The main purpose is to make the model more accurate. As the equation 3.3 [32]:

$$loss = -\sum_{i=1}^{n} \hat{y}_{i1} \log y_{i1} + \hat{y}_{i2} \log y_{i2} + ... + \hat{y}_{im} \log y_{im} \tag{3.3}$$

The following equation is the algorithm used to predict scores in this paper. The new image is imported into the trained model, and this algorithm is used to predict the facial symmetry score.As the Algorithm 1

Algorithm 1 Facial Symmetry Score Algorithm

```
Input : Facial symmetry Dataset
Outout : Facial symmetry score
Result ← Model predictive group
confid ← Model predictive confidence
for i = 1; i < Input.length; i++ do
    if (Result = high level group & confid > 80 ) then
        Output ← high score group score > 8.0;
    else if (Result = high level group & confid < 80 ) then
        Output ← 7.0 > high score group score < 7.9;
    else if (Result = low level group & confid > 80 ) then
        Output ← low score group score < 5.5;
    else if (Result = low level group & confid < 80 ) then
        Output ← 5.7 > low score group score > 7.0;
    end if
end for
```

In addition, our learning rate is between 0.001 and 0.0001. This thesis design the automatic adjustment method to define the learning rate. The purpose is to let the model test in different learning rates to find the best learning rate for training, and this work also Define Early stopping [33], let the model automatically stop training when over-fitting, mainly monitor the validation loss during training. If there is no progress over 10 eopch, it will stop the training of the model and further adjust the model parameters.

3.6 Facial Symmetry Standard

For the standard of facial symmetry, this work cooperated with the doctors of Chang Gung Memorial Hospital in Taiwan, and asked experts to give a score on the images of computed tomography. A total of 8 doctors gave symmetry scores to 71 subjects. Because symmetry does not have standard weights and measures, the scores will vary from person to person. This work averaged it and calculated the standard deviation, and excluded it from the value that is too different from the standard deviation. The facial symmetry of 71 subjects was divided into high-scoring group and low-segment group according to the median as the standard.

Chapter 4

Experimental Results

4.1 Experimental Environment

This experiment uses a physical machine, which is mainly established by a master. Its specifications are displayed in Tale 4.1, and each version number is displayed in 4.2. TensorFlow's GPU version is built in the new version of CUDA.Because the graphics card used must be higher than the CUDA version of the graphics card driver, it must be higher than the officially configured CUDA 9.0.

Table 4.1: Distributed computing environment

	CPU & GPU	RAM	Disk
Master	Intel(R) Core(TM) i7-3970x CPU @ 3.5GHz Nvidia GeForce RTX2080 Ti	64G	1T

Table 4.2: Software specification

Version
Window 10
Anaconda-4.5.12
Python-3.6
CUDA 10.0
Cudnn 7.3
TensorFlow 1.12
Keras-2.2.4
OpenCV-3.4.5

4.2 Image Processing of CT Image

The original data was the CT image file. The original file was a 3D file. This work converted the file into a contour map with 3D features. First, the patient's image is first transferred to Simplant O&O and the CT threshold is adjusted to the soft tissue range. If there is a gap in the skull, it will affect the subsequent cutting results. Therefore, it is necessary to fill the gaps, so fill the gaps in the Mask and create a 3D Object. Next put the head position suitable, and the way to position the head position is as follows

- Put the inner and outer corners of the eyes on the same line (using the average)
- Put the upper edge of the ear hole (feature point PO) of the right ear on the same line as the lower edge of the right eye frame (feature point Or)
- Adjust the position of the eyelid to the same line

After the suitable, the 3D Object is output into an STL file,The process is shown in Figure 4.1, 4.2, 4.3.

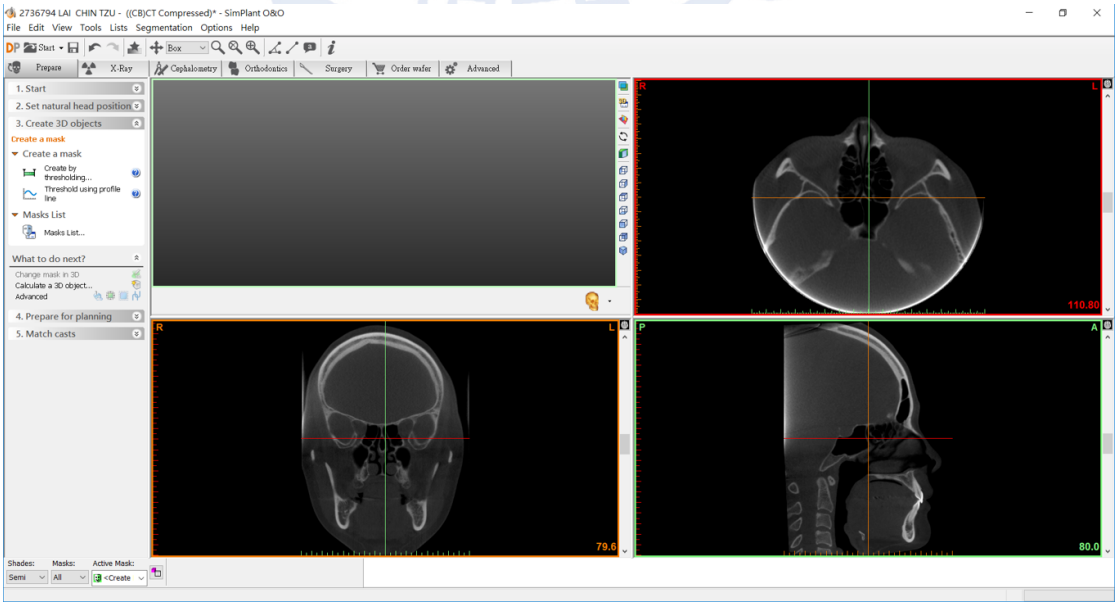


FIGURE 4.1: 3D object after suitable1

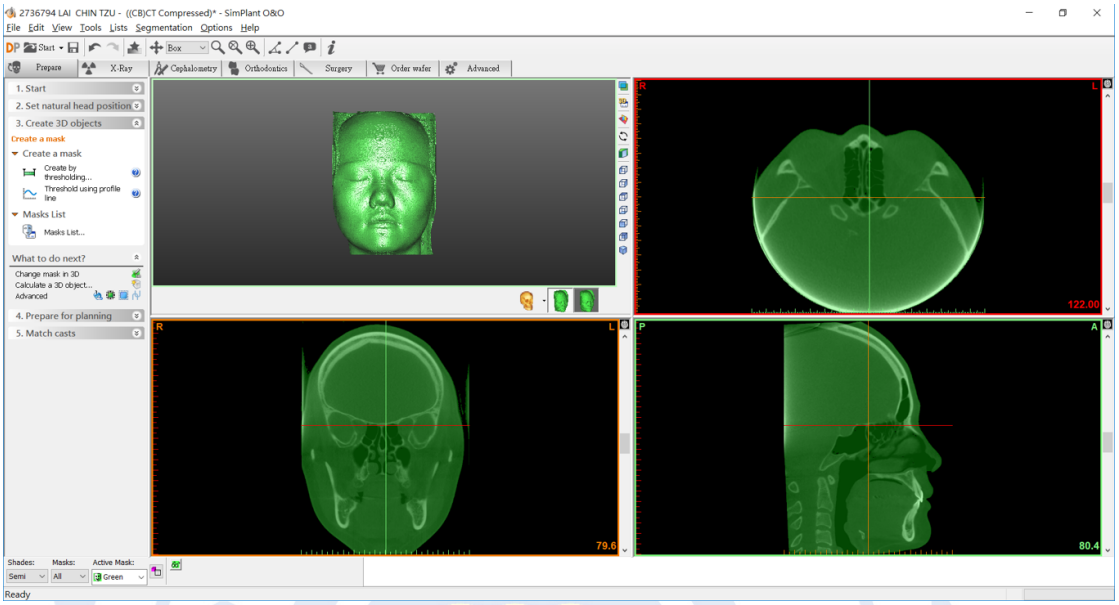


FIGURE 4.2: 3D object after suitable2

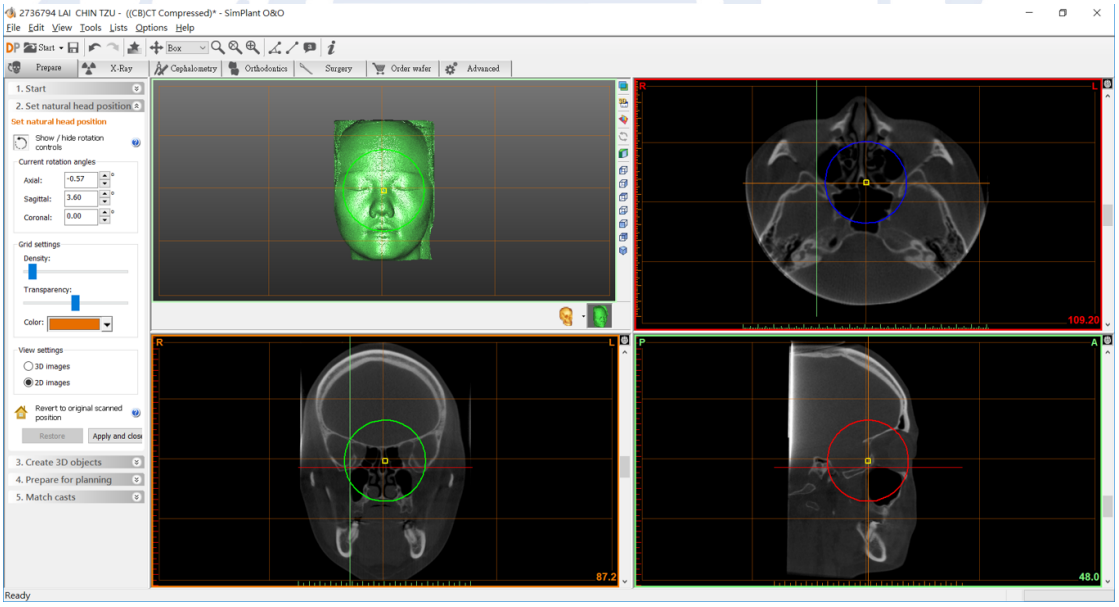


FIGURE 4.3: 3D object after suitable3

Next, this work use Geomagic to tailor our 3D object. The principle of tailoring is the following three points, and before and after clipping, as shown in Figure 4.4

- Excluding noise
- Keep about 3cm above the brow bone (but it will increase or decrease according to the actual situation)
- Remove part of the neck

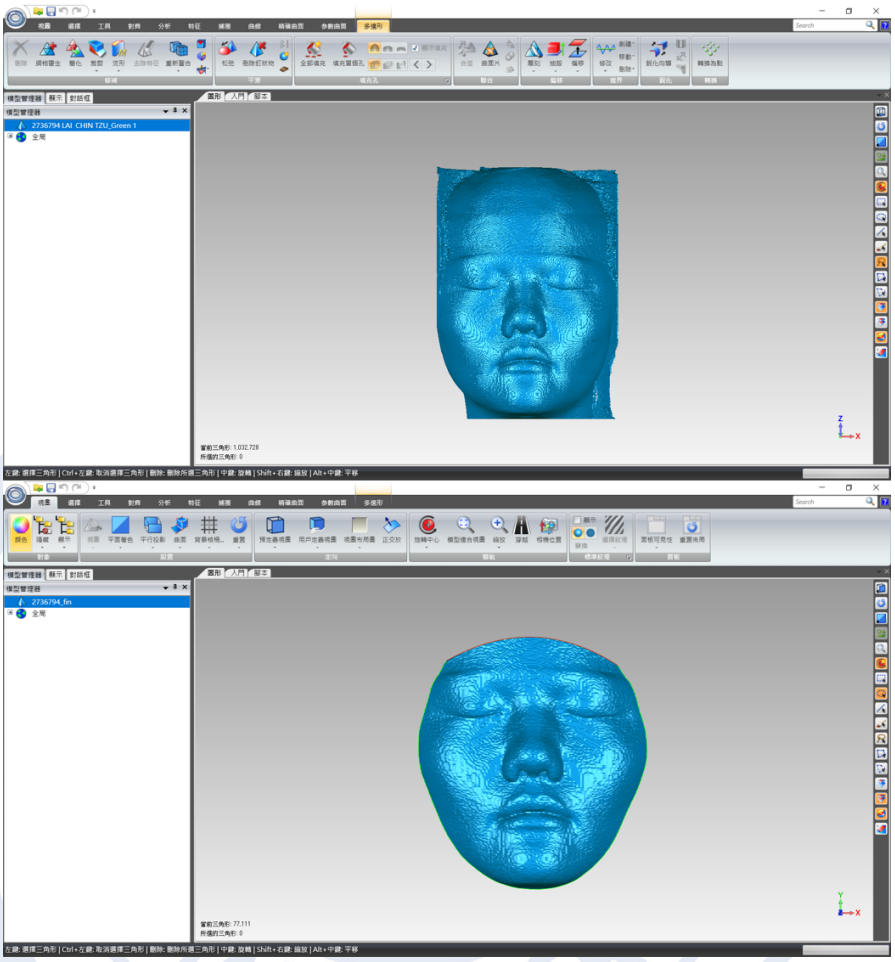


FIGURE 4.4: Before and after clipping

Then convert the cropped 3D Object into points, save the point coordinates into vtx files, and finally use MATLAB to draw the contour lines to complete the contour map used in our training.as shown in Figure 4.5,Figure 4.6 and Figure4.7

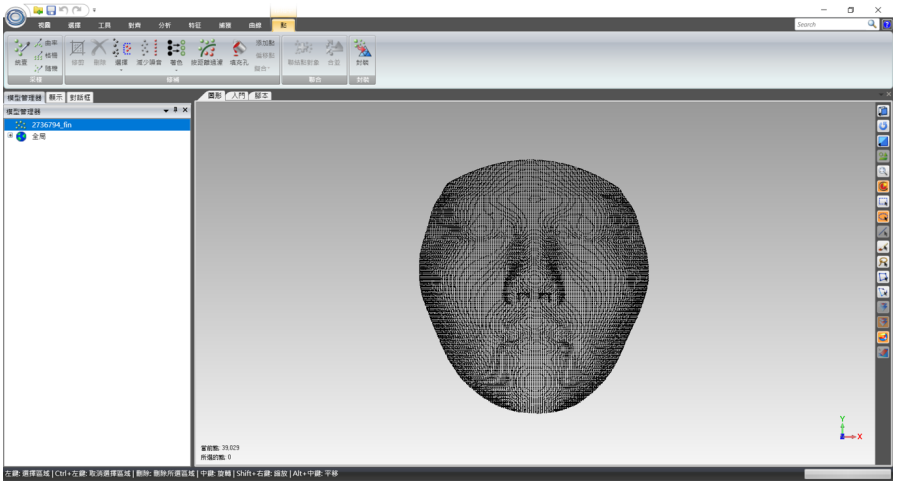


FIGURE 4.5: Convert the cropped 3D object into points

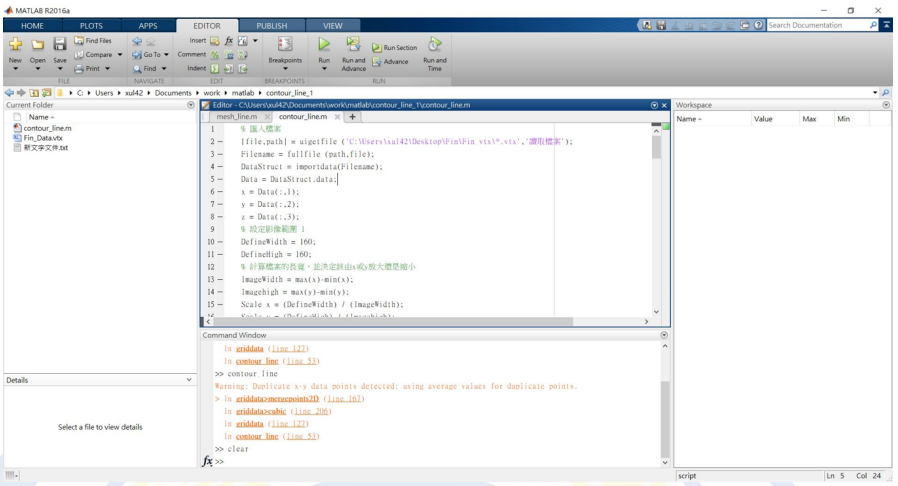


FIGURE 4.6: MATLAB code

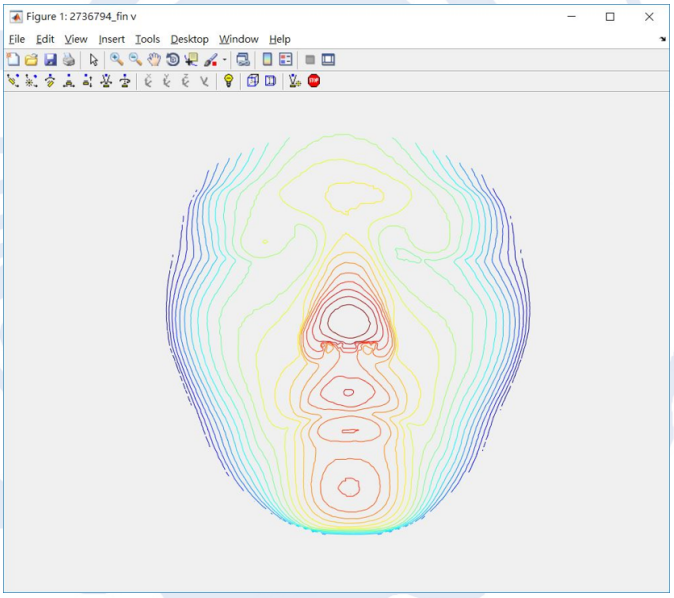


FIGURE 4.7: Contour map

4.3 Differences in the Blank Value Complement

For different pre-processed data, this work use experiments to find the pre-processing data with the best results. In addition to cutting and superimposing the original image, must amplify the original small amount of data. this work use rotation, scaling, and blanks to fill the difference. The main difference in the way of processing is to fill in the blanks. Divided into two categories, the first type of mode is called nearest in Keras, and the data complement is filled with neighboring values.

When our image is scaled or rotated, the blank value generated by the displacement is filled by complementing the neighboring values. The second type of mode is called constant in Keras, and the blank value generated by the displacement is filled with black, as shown in Figure 4.8.Left is constant, Right is nearest.

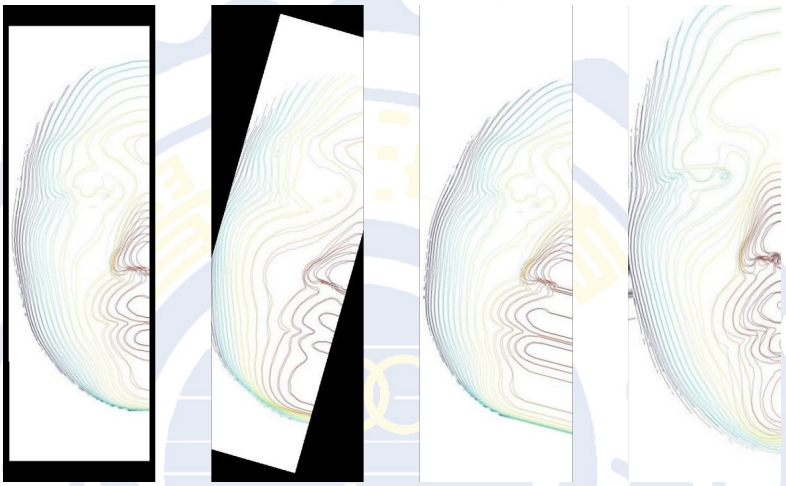


FIGURE 4.8: Differences in the blank value complement

4.4 Verification of True Accuracy

Regarding the verification of true accuracy, our target task is to score facial symmetry using the Transfer Learning training model. There is no objective standard for facial symmetry. it can get very subjective answers when look at the naked eye. Therefore, this work use the scoring criteria of 8 doctors to find out the 6 images of the highest score and the lowest score. This rating data has ruled out excessively large values by excluding values from 1 to 3 standard deviations through the average, which allows the machine to learn the best and worst criteria. The 59 images that the machine has not seen are regarded as the standard of verification, and these images are also scored by doctor. The median score of 71 subjects in our scoring data was 7. So the criterion for our evaluation model is to enter these 59 images to see if the machine can accurately classify it as a high-score or low-score group.

4.5 Transfer Learning for Face Symmetry

Our goal is to use Transfer Learning to train a facial symmetry scoring model that preserves 3D features, so this work use 4 different pre-trained models for training. First, import the pre-processed data through the program. Next need to import the pre-trained model into our new model. Keras can quickly import the pre-trained model and its parameters. The pre-trained models this work used were originally trained on ImageNet. The training set has 1000 classifications, and our target task classification is relatively small. It need to increase the fully connected layer to help convergence. For our small datasets and target tasks with fewer classifications, it can make our models learn our features more quickly. The four models this work used in our experiments are VGG16, VGG19, ResNet50, and Xception. This work compare the pros and cons of using different pre-trained models with the same new model architecture and find the pre-trained model that works best for us.

Table 4.3: New layers detail

GlobalAveragePooling2D()
Dense(1024) activation=relu
Dense(1024) activation=relu
Dense(512) activation=relu
Dense(512) activation=relu
Dense(256) activation=relu
Dropout(0.25)
Dense(128) activation=relu
Dropout(0.25)
Dense(2) activation=softmax

Table 4.4: Model parameter detail

Input Size = 250,750,3
Total Data = 1194 images in 2 class
Batch Size = 4
Epoch = 1000
Steps per epoch = 247
Loss Function = categorical crossentropy

In this experiment, after importing the pre-trained model, the GlobalAveragePooling layer, the 6-layer fully connected layer and 2 Dropout layer, and a

classification layer are added to the model. The detailed layer setting is shown in the table 4.3.The training parameters are respectively set in the table 4.4, and the training has the setting to automatically adjust the Learning Rate, and the adjustment range is set at $0.001 \sim 0.0001$, such as Training accuracy has not improved, more than 3 eopch will automatically adjust. In order to prevent overfitting, this experiment also has the Early stopping, monitoring Validation Loss if more than 10 epoch is not reduced, will stop training the model, and the DataGenerator in Keras will be added to the program to randomly change the data into the model.

4.6 Deep Learning without Fine Tuning

In this experiment, instead of doing fine tuning, the original Pre-trained model architecture and original parameters are directly applied, and a new fully-connected layer and classifier are added. This way is to apply the original feature extraction performance of the model, without using the new data set for fine tuning, but to classify directly. The new data set is adjusted and sorted directly. This method is suitable for use on small data sets. For example, the data set cannot be amplified and the quantity is too small to fine tuning the model. This experiment will apply this method to different models corresponding to different pre-process methods.

4.6.1 Train with Nearest Mode without Fine Tuning

Using the nearest pre-process method to train the model. The dotted line in the figure 4.9 is the training set loss value, and the solid line is the loss value of the verification set. It can be seen from figure that the pre-process method is not efficient in perf*orming fine tuning training. Because our dataset differs greatly from the dataset originally trained by the Pre-trained model, it is not appropriate for our dataset to be applied to this method of training. In addition, due to the pre-process method of nearest, the edge of the image is slightly modified, resulting in the inability of the whole model to converge smoothly. And through this experiment, it can be found that the loss of all model training sets is quickly separated

from the loss of the verification set. The model's early stopping mechanism was activated and the training was stopped at approximately 10 epochs.The dotted line in the figure4.10 is the training set accuracy value, and the solid line is the verification set's accuracy value.It can be observed from figure that the accuracy of the training is neither good for the training set nor the validation set, so this work will not verify the true accuracy of the model.In the following picture, the dotted line is the training set value, and the solid line is the value of the verification set.

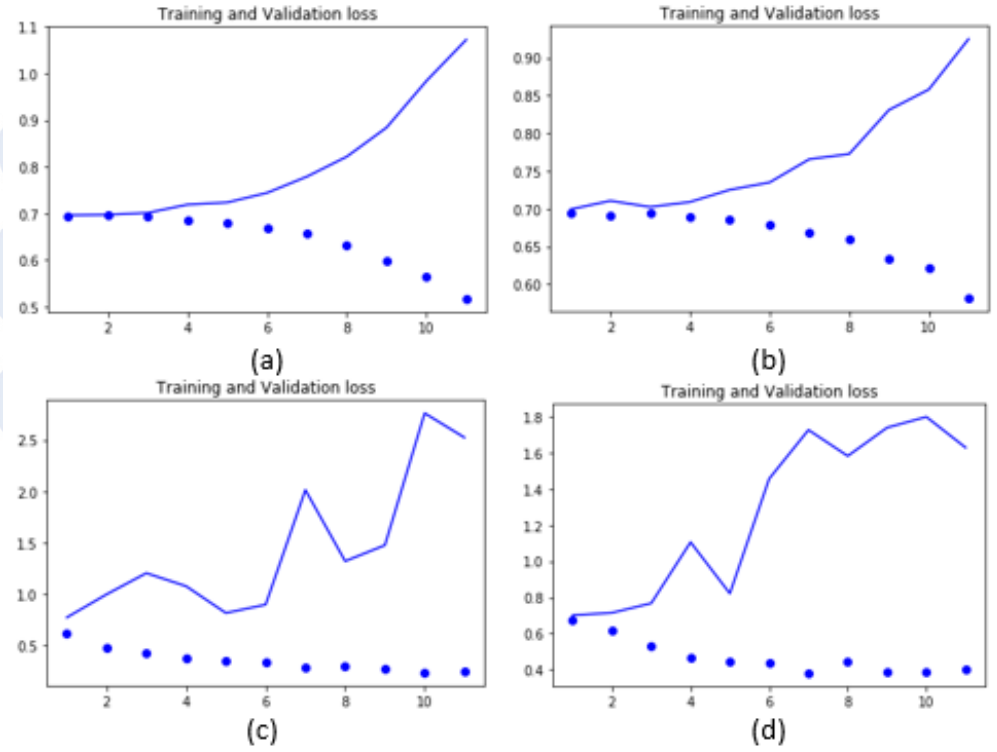


FIGURE 4.9: (a)VGG16, (b)VGG19, (c)ResNet50, (d)Xception nearest mode loss value without fine tuning

4.6.2 Train with Constant Mode without Fine Tuning

Next, this work experimented with the constant pre-process method to train the model. It can be observed from Figure X that the VGG16 model and the VGG19 model were tested in the way this thesis designed, similar to the use of the nearest method, the rapid separation of the loss of the two data sets, This means that this method is not suitable for our dataset, and the Xception model converges

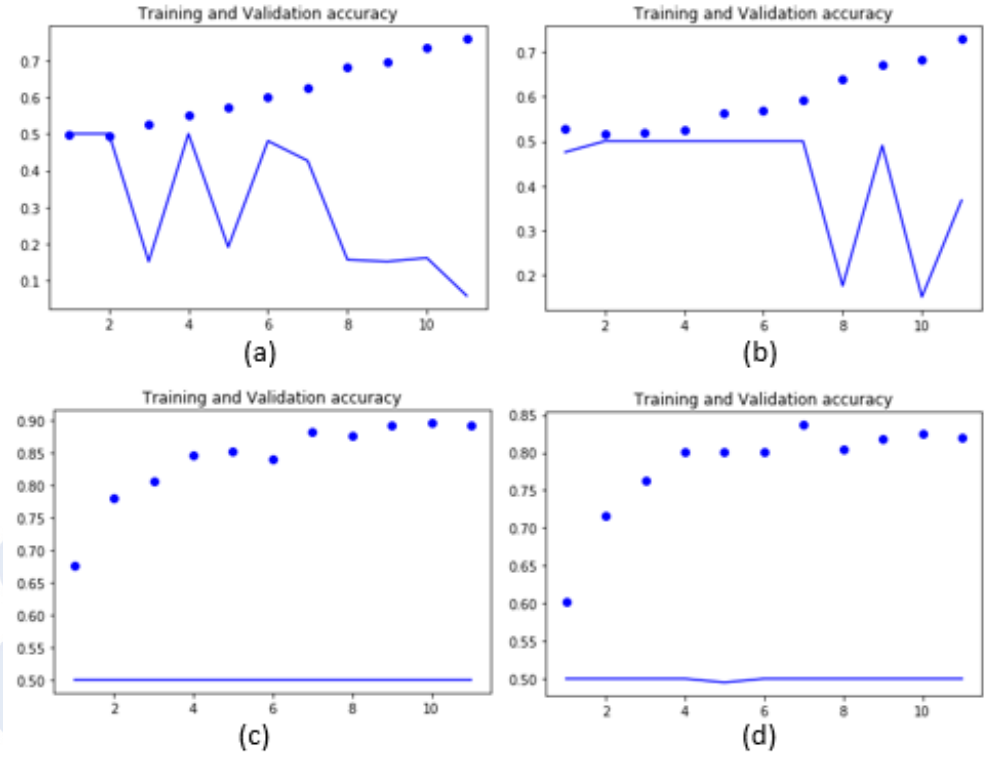


FIGURE 4.10: (a)VGG16, (b)VGG19, (c)ResNet50, (d)Xception nearest mode accuracy without fine tuning

better than the Resnet50 model and the Xception model. But overall, the four models are not able to train smoothly. The accuracy is quite low so no further verification accuracy is done. It seems from both experiments that no matter what pre-process method. If you do not fine tuning the parameters of the whole model during training, the training performance is not good, and it is not suitable for the architecture and data set this thesis designed.Figure 4.12 shows the accuracy of the training set and validation set of the model during training.

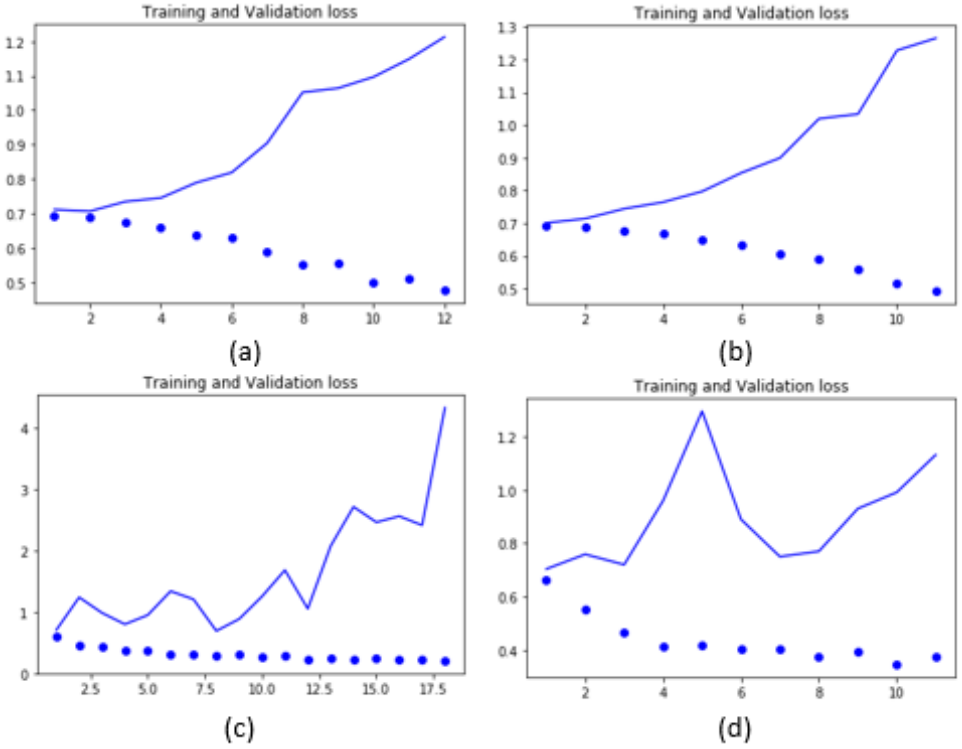


FIGURE 4.11: (a)VGG16, (b)VGG19, (c)ResNet50, (d)Xception constant mode loss value without fine tuning

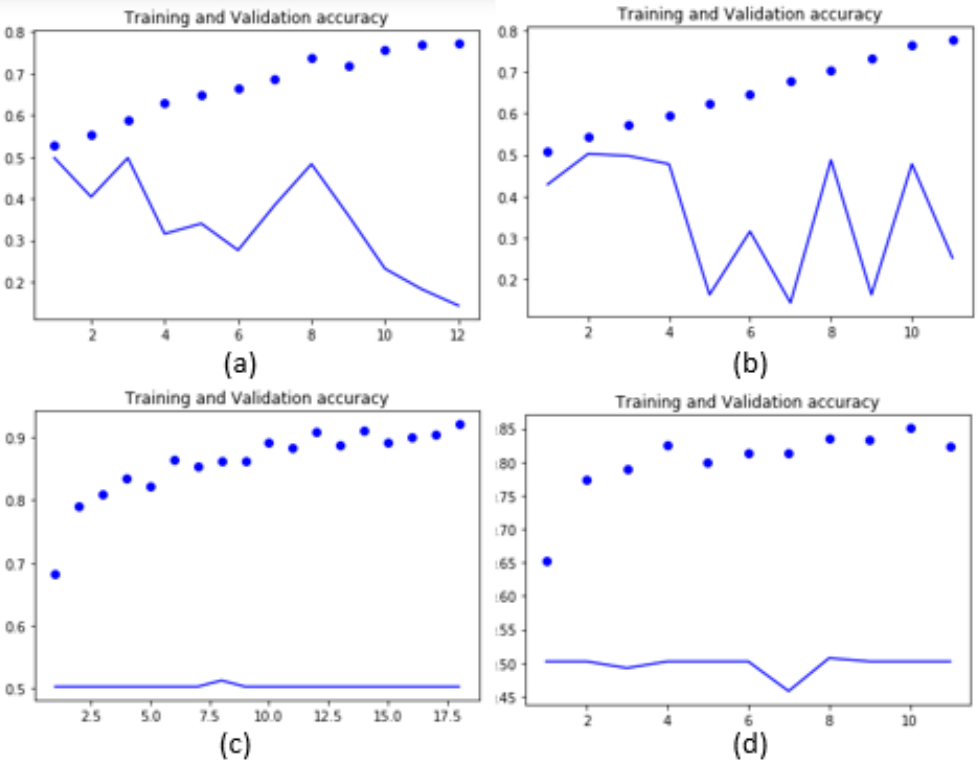


FIGURE 4.12: (a)VGG16, (b)VGG19, (c)ResNet50, (d)Xception constant mode accuracy without fine tuning

4.7 Deep Learning Fine Tuning

This experiment is to make a fine tuning of the Pre-trained model. Use our dataset to fine tune the model parameters. The goal is to have this model train in a data set that has not been seen, and to be able to more accurately distinguish the target through fine-tunable parameters. However, this method is suitable for use when the data set has a certain amount. It is usually applied to a small number of data sets and cannot be trained in the new deep learning model architecture. By using Transfer Learning, this data set is not the same as the original Pre-trained model. This method will be used to fine tune the model and train the Pre-trained model to accurately classify new data sets.

4.7.1 Train with Nearest Mode with Fine Tuning

In this experiment, the pre-process method is used in four kinds of Pre-trained models, which can be seen from the above figure 4.8. Therefore, it can be observed that the picture is slightly deformed. The larger the rotation of the figure is, the more severe the deformation is. However, the model this work learning is the symmetry of the face, and the features of the superimposed picture are not eliminated. Therefore, this work did this experiment to verify the pros and cons of this pre-process method for model learning.

It can be observed from the experiment that, as shown in Fig4.13, the Loss values of VGG16 and VGG19 cannot be smoothly reduced. And can't successfully learn good features, it can't effectively distinguish the difference between high-score group and low-score group. it can be found that VGG16 could not effectively reduce the loss value of the verification set, and it was Early stopping on very few epochs. The VGG19, ResNet50, and Xception models are both undulating and cannot effectively converge the loss value to near zero. And you can see that all the models used the Nearest pre-precess method to train the results are not excellent, all around 30 epochs due to performance stagnation and Early stopping to stop training.

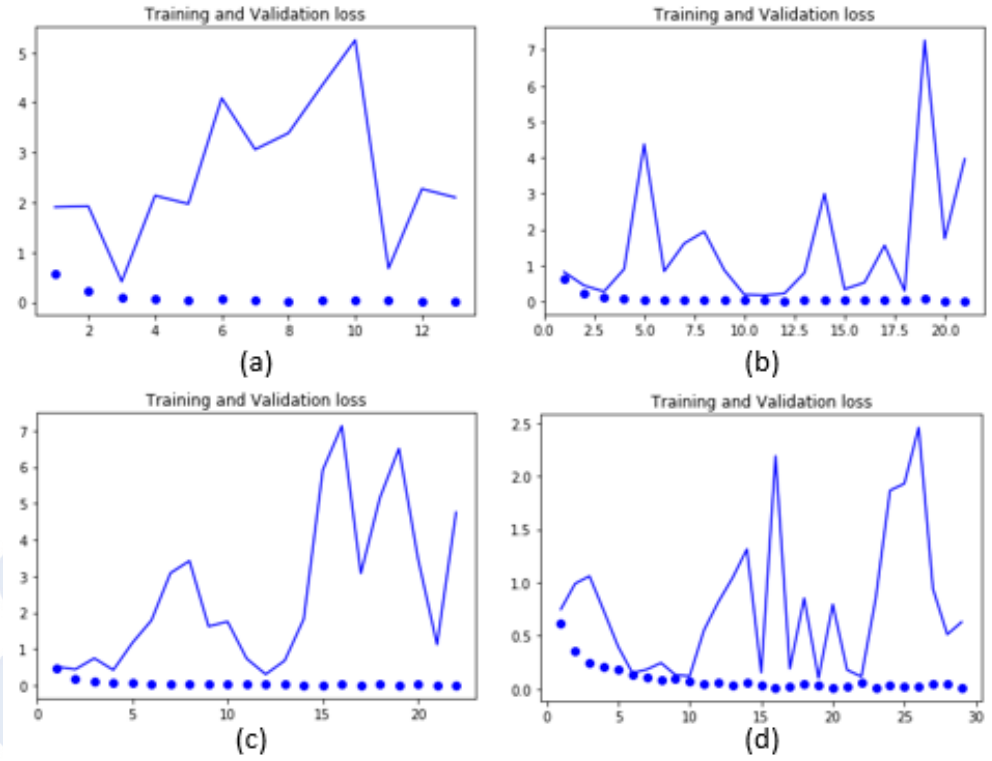


FIGURE 4.13: (a)VGG16, (b)VGG19, (c)ResNet50, (d)Xception nearest mode loss value

Figure 4.14 shows the accuracy of the training set and validation set of the model during training. It can be observed that the accuracy of the four model training sets is 100%, but the accuracy of the validation set is very fluctuating, indicating that the verification is not as accurate as the training. Below will verify the true accuracy.

Next step have to verify the true accuracy,In the following table, calculate the true accuracy of the model through the program. Table 4.5 shows that the VGG16, ResNet50, and Xception models all achieve more than 70% accuracy, while VGG19 only achieves 66% accuracy in our design method. It can be observed from the experiment that although VGG16 triggers Early stopping at an earlier time due to performance stagnation, the performance is relatively better than VGG19. Since our experiments are all based on adding the same model architecture to Pre-trained models and exploring the best method in this architecture, it is possible that VGG16 is better than VGG19 for this reason.

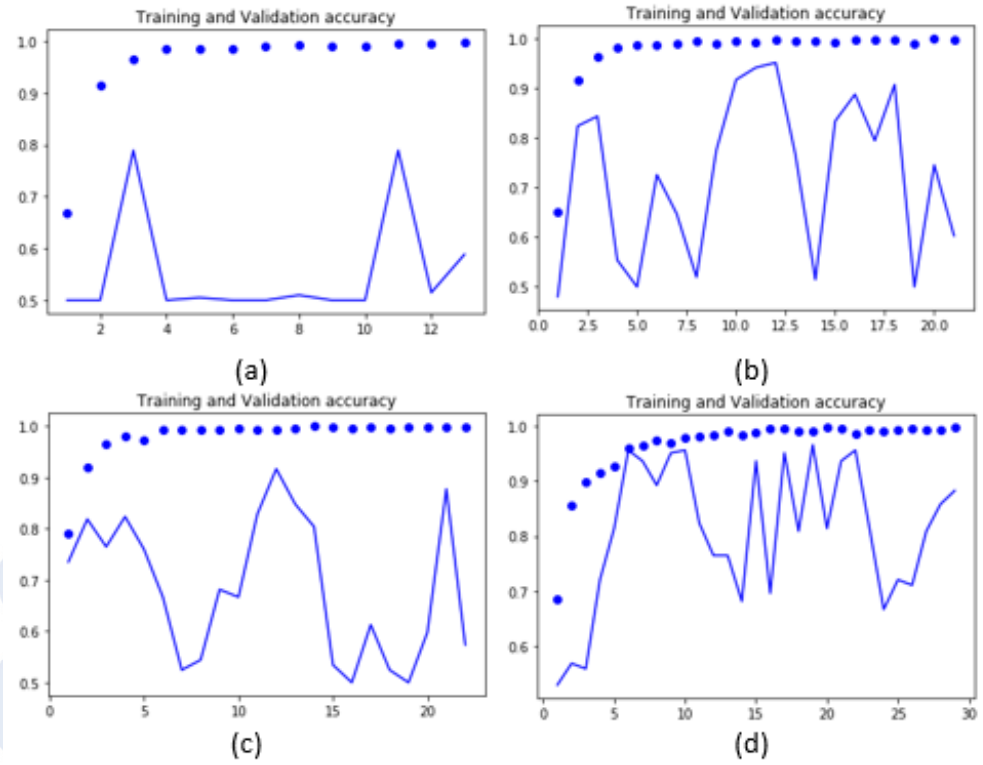


FIGURE 4.14: (a)VGG16, (b)VGG19, (c)ResNet50, (d)Xception nearest mode accuracy

TABLE 4.5: Nearest mode accuracy between Xception and ResNet50

Model	True	False	Total	Accuracy
VGG16	43	16	59	73%
VGG19	39	20	59	66%
ResNet50	42	17	59	71%
Xception	45	14	59	76%

4.7.2 Train with Constant Mode with Fine Tuning

In this experiment, the pre-process method used in the Pre-trained model is constant. It can be observed from the above figure4.8 that the method does not change the pattern, but maintains the pattern as it is but fills the black space in the blank. Therefore, the graphics will not be pulled to cause changes. It can be observed from the experimental results that VGG16 and VGG19 cannot successfully decline the Loss value in this task. Unable to successfully train learning

features. On the other hand, ResNet50 and Xception, they successfully converge quickly, and the Loss value drops to near zero. It can be found from experiments that Xception continues to converge in a stable state, and more than doubles that of other model epochs. The model stops because the performance has no progress, but the relative fluctuations are large, Xception and ResNet-50 are successful. Convergence, this work will further verify the true accuracy rate and will explore the issue of verification accuracy.As shown in Figure 4.15

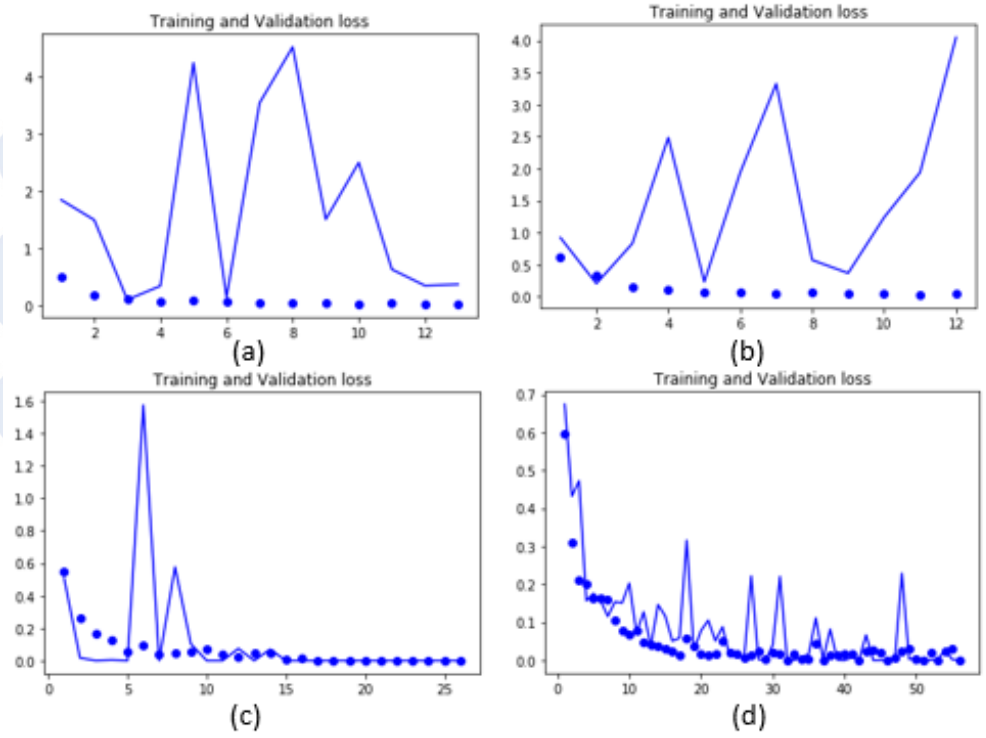


FIGURE 4.15: (a)VGG16, (b)VGG19, (c)ResNet50, (d)Xception constant mode loss value

The results of this experiment show that the overall accuracy is better than the method using Nearest pre-process. The VGG16 model improved by 7%, VGG19 achieved a very large improvement of 20%, ResNet50 improved by 12%, and Xception improved by 14%. In addition, it can be found that the training process of VGG19 was not very good, and overfitting occurred at the end of training. Although the training loss has approached zero, the validation loss has increased. In this case, the accuracy of the model verification is still 86% higher than the VGG16 model and the ResNet50 model. Although the training process of ResNet50 is very

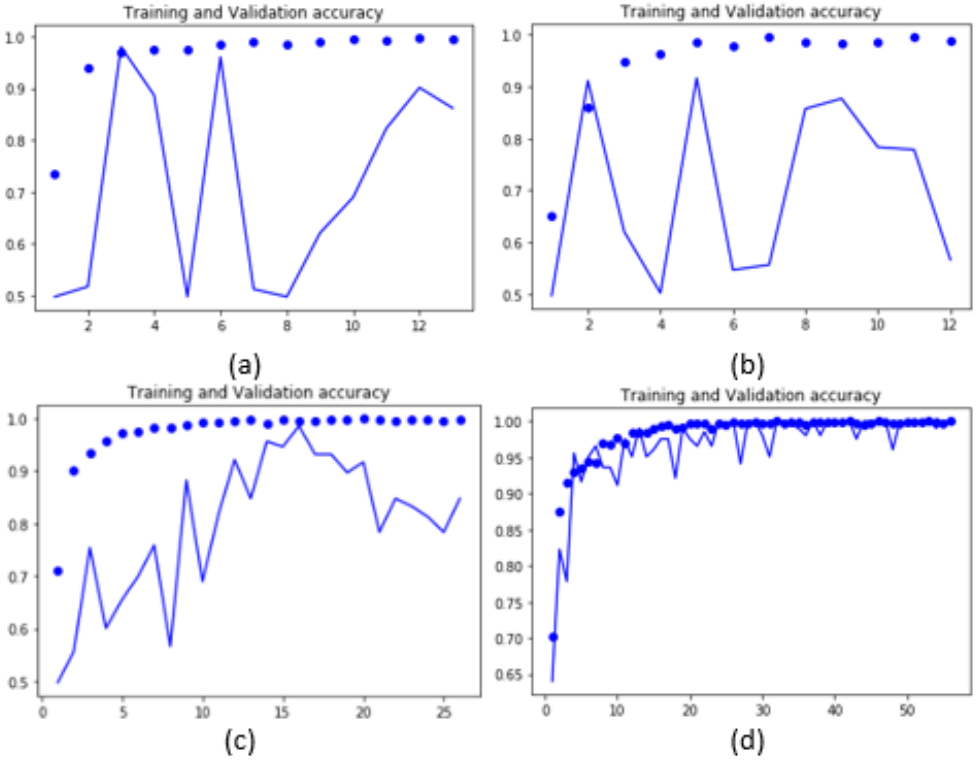


FIGURE 4.16: (a)VGG16, (b)VGG19, (c)ResNet50, (d)Xception constant mode accuracy

smooth, the true accuracy rate of the model is only 83%, which may result in poor characteristics during the training. The Xception model is the best in the four models, and the training process is the smoothest, but it can be found that even with such a smooth training, this experiment can not achieve 100% accuracy, so next paragraph will explore the reasons. As shown in Table 4.6

TABLE 4.6: Constant mode accuracy between Xception and ResNet50

Model	True	False	Total	Accuracy
VGG16	47	12	59	80%
VGG19	51	8	59	86%
ResNet50	49	10	59	83%
Xception	53	6	59	90%

In the process of verifying the best Xception model, udged the wrong 6 data to explore the reasons why 100% accuracy could not be achieved. It can be found from Figure 4.17 that the data of the three judgment errors are all low-group

erroneously judged as high-group. In the figure, a is scored 6.8, b is 6.7, and c is 6.5. The reason why these pieces of data are low-segmented is that the asymmetrical positions are at the edge of the chin. It can be found that the images misjudged by the Xception model are of this type. The score ranged from 1 to 10 points, but the data balance for our data set was based on the median grouping criteria. The median score of the doctor's score in our data set was 7. The score distribution is shown in Figure 4.18 (the X-axis is the total number of data sets and the Y-axis is the rating of 8 doctors). Therefore the median is biased toward high scores, causing the model to misjudge it as a high score when judging the value near the midline. This is also the case that our model is misclassified as high-segment in 6.5∼7.

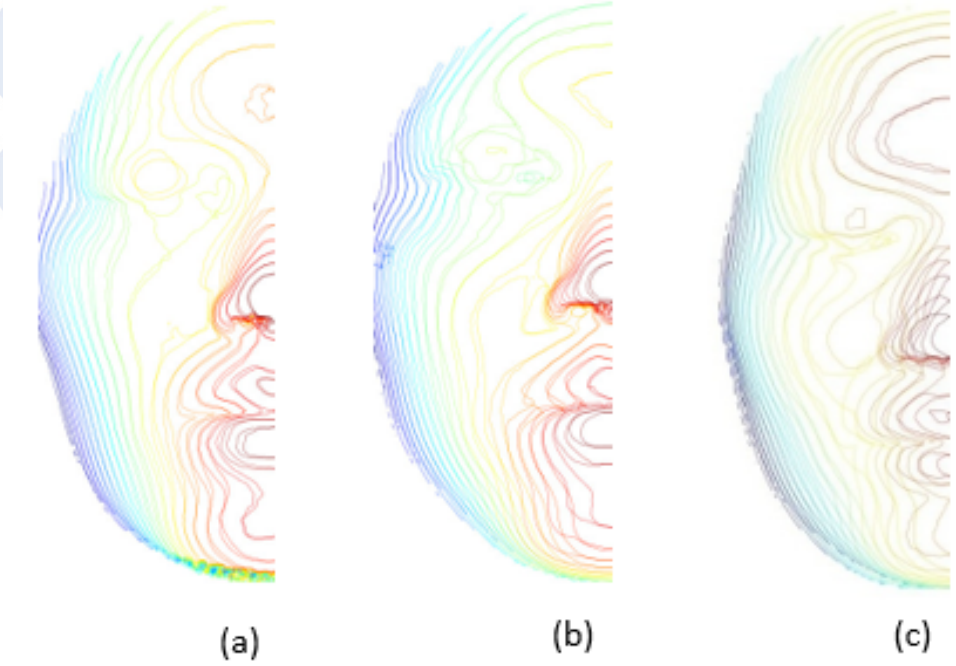


FIGURE 4.17: Discussion on the picture of model identification error

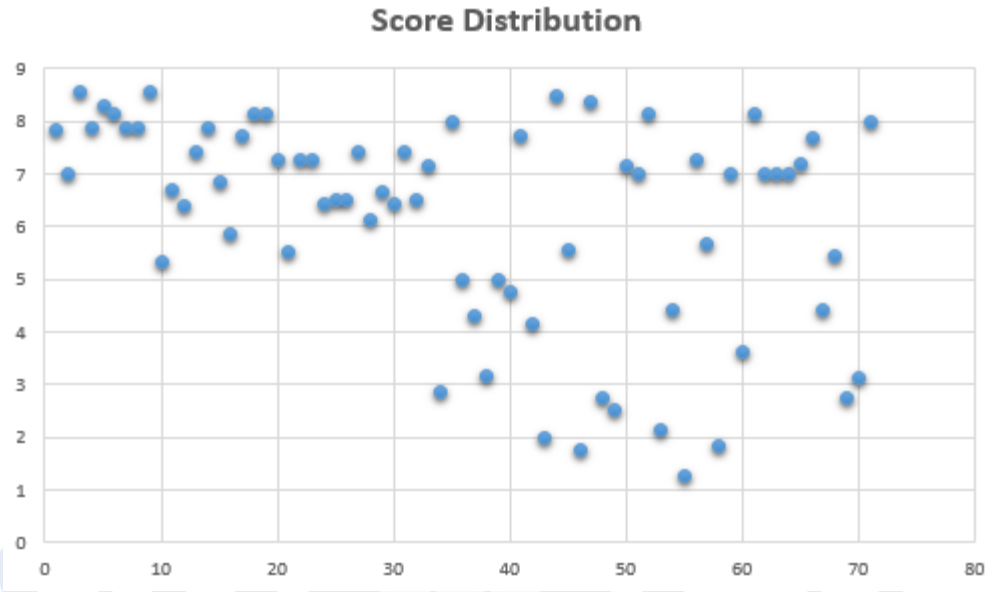


FIGURE 4.18: Score distribution map

4.8 Experimental summary

4.8.1 Prediction

Our goal is to evaluate the symmetry of the face, so need to generate a score for the new face through the trained model. Use the confidence value given by the model to identify the classification as the indicator, and give the corresponding score based on the size of the confidence value. Set the model's scoring criteria to be based on the average of 8 doctors' criteria for giving us training data sets and excluding the differences. For example, the model gives a higher confidence value for high scores and approaches 100%. Give a relatively high score through this confidence value, and vice versa. If the confidence value given by the model is biased towards low scores, the corresponding ones will be predicted. fraction. In this way, it can predict the facial symmetry score of the new data through the trained model. And the score in this way is more objective than the score of a single doctor.Figure 4.19 is the score predicted by the model. The picture on the left is a high score picture and the model is predicted to be 7.6 points. On the right is a low score picture, the model is predicted to be 5.3 points.

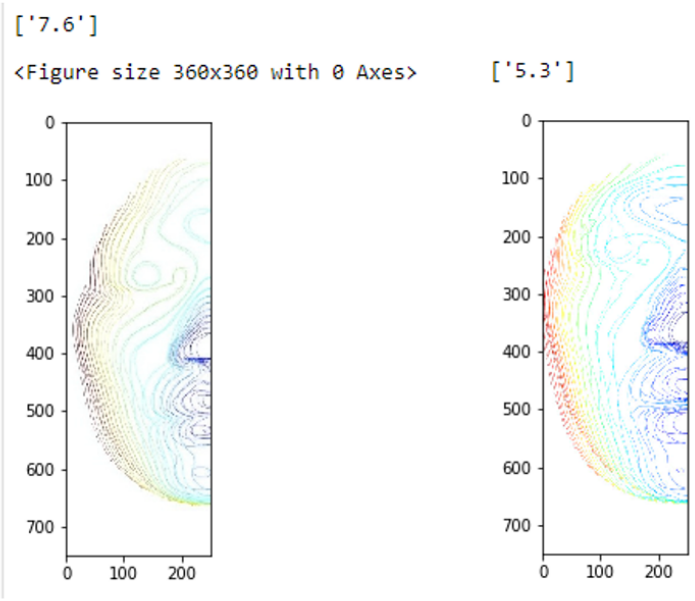


FIGURE 4.19: Model prediction

4.8.2 Total Experiment Comparison

This part draw a comparison chart of all our experimental results, as shown in Figure 4.20. This chart shows the four training methods and their accuracy in our experiments. The accuracy ratios of Nearest without fine tuning and Constant without fine tuning are the accuracy of the test set during training. this thesis do not verify the true accuracy of the model due to poor training results. Nearest with fine tuning and Constant with fine tuning are the true accuracy of the verification. Through all the experiments, it can be found that the pre-process Constant method is suitable for our target tasks, and the model used for the Transfer Learning training is more suitable for using the Xception model.

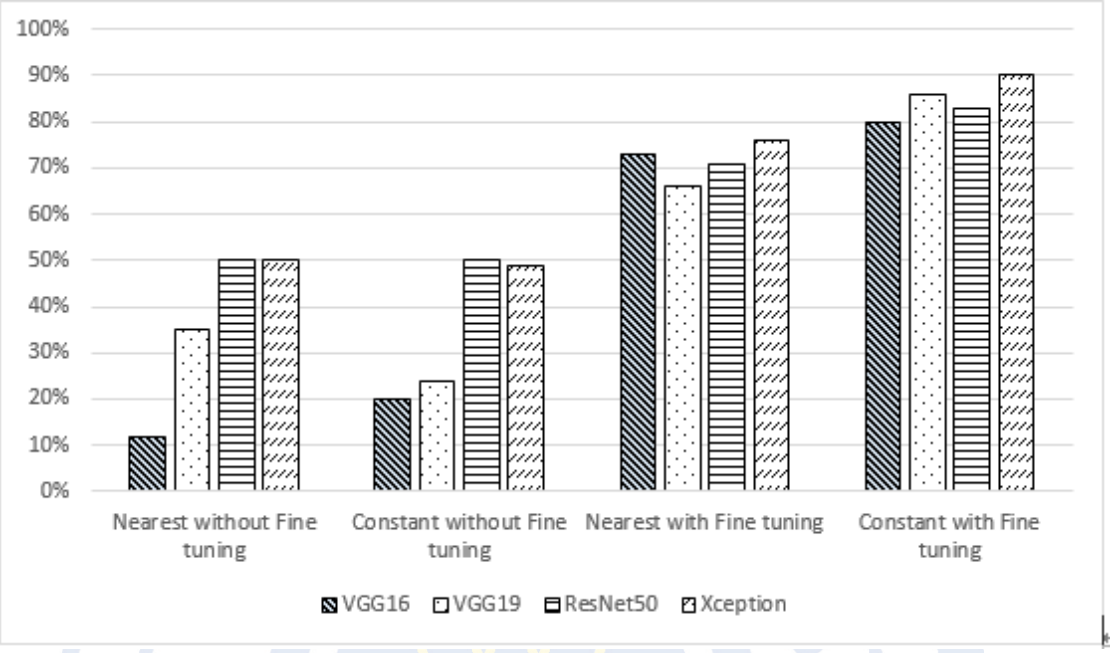


FIGURE 4.20: Summary comparison

Chapter 5

Conclusions and Future Works

5.1 Conclusions

This work used Transfer Learning to evaluate facial symmetry before and after orthognathic surgery, and through Transfer Learning to capture features of the Pre-Trained model with powerful features. this thesis added a neural layer to the model to improve the accuracy of our new model and allow the Pre-Trained model to learn better features. MATLAB processed CT scans into facial contour maps. In this thesis designed experiments to compare different data pre-process methods and experiments with different Pre-Trained models. It has been found through experiments that the data amplification method uses Constant to perform better. The Xception model can learn better features, and the true accuracy is the best. Through the results of the experiment, this thesis finally chose the Xception model with Constant's data amplification method, combined with the new neural layer and dataset adjustment. It is applied to the evaluation of facial symmetry before and after orthognathic surgery. The trained new model allows for effective facial symmetry assessment.

5.2 Future Works

In the future, our goal is to collect a larger number of facial data sets and design a better neural network architecture to make our models more accurate. Compare the pros and cons of more different Pre-Trained models, and develop our own models with new data and add comparisons. Expect more refined results to further explore the possibilities of more medical imaging related applications.

References

[1] R. Anderson, A. P. Gema, , and S. M. Isa. Facial attractiveness classification using deep learning. In *2018 Indonesian Association for Pattern Recognition International Conference (INAPR)*, pages 34–38, Sep. 2018.

[2] Wen-Chung Chiang, Hsiu-Hsia Lin, Chiung-Shing Huang, Lun-Jou Lo, and Shu-Yen Wan. The cluster assessment of facial attractiveness using fuzzy neural network classifier based on 3d moiré features. *Pattern Recognition*, 47(3):1249 – 1260, 2014. Handwriting Recognition and other PR Applications.

[3] Gillian Rhodes. The evolutionary psychology of facial beauty. *Annual Review of Psychology*, 57(1):199–226, 2006. PMID: 16097897.

[4] Judith H. Langlois and Lori A. Roggman. Attractive faces are only average. *Psychological Science*, 1(2):115–121, 1990.

[5] A.J O'Toole, T Price, T Vetter, J.C Bartlett, and V Blanz. 3d shape and 2d surface textures of human faces: the role of "averages" in attractiveness and age. *Image and Vision Computing*, 18(1):9 – 19, 1999.

[6] David I Perrett, D.Michael Burt, Ian S Penton-Voak, Kieran J Lee, Duncan A Rowland, and Rachel Edwards. Symmetry and human facial attractiveness. *Evolution and Human Behavior*, 20(5):295 – 307, 1999.

[7] Wikipedia contributors. Python (programming language) — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Python_(programming_language)&oldid=892641634`, 2019. [Online; accessed 16-April-2019].

[8] Wikipedia contributors. Tensorflow — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=TensorFlow&oldid=892334274`, 2019. [Online; accessed 16-April-2019].

[9] Keras. Keras: The python deep learning library, 2015. [Online; accessed 2018].

[10] Wikipedia contributors. Convolutional neural network — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Convolutional_neural_network&oldid=891636189`, 2019. [Online; accessed 16-April-2019].

[11] Wikipedia contributors. Transfer learning — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Transfer_learning&oldid=891570550`, 2019. [Online; accessed 16-April-2019].

[12] Alessandra Lumini and Loris Nanni. Deep learning and transfer learning features for plankton classification. *Ecological Informatics*, 51:33 – 43, 2019.

[13] Qing Wang, Panfei Du, Jingyu Yang, Guohua Wang, Jianjun Lei, and Chunping Hou. Transferred deep learning based waveform recognition for cognitive passive radar. *Signal Processing*, 155:259 – 267, 2019.

[14] Rupali Sandip Kute, Vibha Vyas, and Alwin Anuse. Component-based face recognition under transfer learning for forensic applications. *Information Sciences*, 476:176 – 191, 2019.

[15] Luis H.S. Vogado, Rodrigo M.S. Veras, Flavio. H.D. Araujo, Romuere R.V. Silva, and Kelson R.T. Aires. Leukemia diagnosis in blood slides using transfer learning in cnns and svm for classification. *Engineering Applications of Artificial Intelligence*, 72:415 – 422, 2018.

[16] Gianluigi Ciocca, Paolo Napoletano, and Raimondo Schettini. Cnn-based features for retrieval and classification of food images. *Computer Vision and Image Understanding*, 176-177:70 – 77, 2018.

[17] Xinglong Liu, Fei Hou, Hong Qin, and Aimin Hao. Multi-view multi-scale cnns for lung nodule type classification from ct images. *Pattern Recognition*, 77:262 – 275, 2018.

[18] Yuzhu Ji, Haijun Zhang, and Q.M. Jonathan Wu. Salient object detection via multi-scale attention cnn. *Neurocomputing*, 322:130 – 140, 2018.

[19] Shuai Li, Wenfeng Song, Hong Qin, and Aimin Hao. Deep variance network: An iterative, improved cnn framework for unbalanced training datasets. *Pattern Recognition*, 81:294 – 308, 2018.

[20] Qi Xu, Ming Zhang, Zonghua Gu, and Gang Pan. Overfitting remedy by sparsifying regularization on fully-connected layers of cnns. *Neurocomputing*, 328:69 – 74, 2019. Chinese Conference on Computer Vision 2017.

[21] Dongmei Han, Qigang Liu, and Weiguo Fan. A new image classification method using cnn transfer learning and web data augmentation. *Expert Systems with Applications*, 95:43 – 56, 2018.

[22] Vojtěch Franc and Jan Čech. Learning cnns from weakly annotated facial images. *Image and Vision Computing*, 77:10 – 20, 2018.

[23] Wikipedia contributors. Cuda — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=CUDA&oldid=892701084`, 2019. [Online; accessed 17-April-2019].

[24] Nvidia Cudnn. Nvidia developer cudnn. `https://developer.nvidia.com/cudnn`, 2019. [Online; accessed 17-April-2019].

[25] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv e-prints*, page arXiv:1409.1556, Sep 2014.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv e-prints*, page arXiv:1512.03385, Dec 2015.

[27] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. *arXiv e-prints*, page arXiv:1610.02357, Oct 2016.

[28] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[29] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015.

[30] Wikipedia contributors. Rectifier (neural networks) — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Rectifier_(neural_networks)&oldid=890629677`, 2019. [Online; accessed 16-April-2019].

[31] Wikipedia contributors. Softmax function — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Softmax_function&oldid=890873030`, 2019. [Online; accessed 16-April-2019].

[32] ITREAD01. categorical crossentropy. `https://www.itread01.com/content/1543994346.html`, 2018. [Online; accessed2018].

[33] Wikipedia contributors. Early stopping — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Early_stopping&oldid=883916323`, 2019. [Online; accessed 17-April-2019].

Appendix A

Keras with TensorFlow

I. Anaconda Install

```
https://www.anaconda.com/
```

II. Create environment

```
# conda create  -n cnn python=3.6
# activate cnn
```

III. Cuda10.0 Install

```
https://developer.nvidia.com/cuda-downloads
```

IV. Cudnn 7.4 Install

```
https://developer.nvidia.com/cudnn
```

V. TensorFlow GPU 1.12 install

```
download from https://github.com/fo40225/tensorflow-windows-wheel
# pip install tensorflow_gpu-1.12.0-cp36-cp36m-win_amd64.whl
```

VI. Package install

```
# pip install matplotlib pillow pandas scikit-learn opencv-python jupyter
```

VII. Keras Install

```
# pip install keras
```

Appendix B

Data Preprocess Code

I. CutTheLeftHalf.ipynb Code

```
import os, random
import cv2
import numpy as np

fileDir = "C:/Users/HPC/Desktop/true/"
fileDir1 = "C:/Users/HPC/Desktop/a/"
pathDir = os.listdir(fileDir)
for j in range (len(pathDir)):
print(fileDir+pathDir[j])
img = cv2.imread(fileDir+pathDir[j])
x = 0
y = 0
w = 500
h = 750
crop_img = img[y:y+h, x:x+w]
cv2.imwrite(fileDir1+pathDir[j], crop_img)
```

II. CutTheRightHalf.ipynb Code

```
import os, random
import cv2
import numpy as np

fileDir = "C:/Users/HPC/Desktop/true/"
fileDir1 = "C:/Users/HPC/Desktop/b/"
```

```
pathDir = os.listdir(fileDir)
for j in range (len(pathDir)):
print(fileDir+pathDir[j])
img = cv2.imread(fileDir+pathDir[j])
x = 500
y = 0
w = 500
h = 750
crop_img = img[y:y+h, x:x+w]
cv2.imwrite(fileDir1+pathDir[j], crop_img)
```

III. FlipTheRightHalfHorizontally.ipynb Code

```
import os, random
import cv2
import numpy as np

fileDir = "C:/Users/HPC/Desktop/b/"
fileDir1 = "C:/Users/HPC/Desktop/b/"
pathDir = os.listdir(fileDir)
for j in range (len(pathDir)):
print(fileDir+pathDir[j])
img = cv2.imread(fileDir+pathDir[j])
h_flip = cv2.flip(img, 1)
cv2.imwrite(fileDir1+pathDir[j], h_flip)
```

IV. FusionofLeftandRightSides.ipynb Code

```
import os, random
import cv2
import numpy as np

fileDir = "C:/Users/HPC/Desktop/a/"
fileDir1 = "C:/Users/HPC/Desktop/b/"
fileDir2 = "C:/Users/HPC/Desktop/c/"
pathDir = os.listdir(fileDir)
for j in range (len(pathDir)):
print(fileDir+pathDir[j])
img = cv2.imread(fileDir+pathDir[j])
img1 = cv2.imread(fileDir1+pathDir[j])
img_mix = cv2.addWeighted(img, 0.5, img1, 0.5, 0)
cv2.imwrite(fileDir2+pathDir[j], img_mix)
```

V. DataAmplification.ipynb Code

```
from keras.preprocessing.image import ImageDataGenerator, array_to_img, img_to_array, load_img
from os import listdir
from os.path import isfile, isdir, join

datagen = ImageDataGenerator(
zca_whitening=True,
rotation_range=20,
width_shift_range=0,
height_shift_range=0,
shear_range=0,
zoom_range=0.2,
horizontal_flip=False,
fill_mode='nearest')

mypath = "C:/Users/HPC/Desktop/full2/train/1/"

files = listdir(mypath)

for f in files:
img = load_img(mypath+f)
x = img_to_array(img)
x = x.reshape((1,) + x.shape)
i = 0
for batch in datagen.flow(x, batch_size=1,
save_to_dir=mypath, save_prefix='1', save_format='jpg'):
i += 1
print(f)
if i > 100:
break
```

Appendix C

Model Without Fine Tuning Training Code

II. VGG16.ipynb Code

```
import tensorflow as tf
import numpy as np
import gc
import matplotlib.pyplot as plt
import pandas as pd
import time
from PIL import Image
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
from keras.layers import Flatten, Dense ,Dropout
from keras.models import Sequential, Model
from sklearn.model_selection import train_test_split

from keras.preprocessing import image
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint
from keras.applications.xception import preprocess_input


from keras.applications.vgg16 import VGG16
base_model = VGG16(input_shape=(250,750, 3), weights='imagenet', include_top=False)

for layer in base_model.layers[:]:
layer.trainable = False
```

```
# Check the trainable status of the individual layers
for layer in base_model.layers:
print(layer, layer.trainable)



x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
x = Dense(1024, activation='relu')(x)
x = Dense(512, activation='relu')(x)
x = Dense(512, activation='relu')(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.25)(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.25)(x)
predictions = Dense(2, activation='softmax')(x)

model = Model(base_model.input, predictions)
print(model.summary())

from keras.optimizers import *
from keras.callbacks import ReduceLROnPlateau

import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

base_dir = 'C:/Users/HPC/Desktop/full'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'test')

train_datagen = ImageDataGenerator(
rescale=1./255,
rotation_range=30,
width_shift_range=0.3,
height_shift_range=0.3,
zoom_range=0.3,
horizontal_flip=True,
)

validation_datagen = ImageDataGenerator(rescale=1./255)



train_batchsize = 4
val_batchsize = 4

train_generator = train_datagen.flow_from_directory(
train_dir,
```

```
target_size=(250, 750),
batch_size=train_batchsize,
class_mode='categorical'

)

validation_generator = validation_datagen.flow_from_directory(
validation_dir,
target_size=(250, 750),
batch_size=val_batchsize,
class_mode='categorical',
shuffle=False)
print(train_generator.class_indices)

from keras import optimizers
model.compile(loss='categorical_crossentropy',
optimizer=optimizers.RMSprop(lr=0.0001),
metrics=['acc'])
learning_rate_reduction = ReduceLROnPlateau(monitor='acc',
patience=3,
verbose=1,
factor=0.5,
min_lr=0.00001)
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=2)

history = model.fit_generator(
train_generator,
steps_per_epoch=train_generator.samples/train_generator.batch_size ,
epochs=1000,
validation_data=validation_generator,
validation_steps=validation_generator.samples/validation_generator.batch_size,
class_weight='auto',
verbose=1,
callbacks=[learning_rate_reduction,early_stopping])

import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label="Training acc")
plt.plot(epochs, val_acc, 'b', label='Validation_acc')
plt.title('Training and Validation accuracy')
```

```
plt.legend

plt.figure()

plt.plot(epochs, loss, 'bo', label="Training loss")
plt.plot(epochs, val_loss, 'b', label='Validation_loss')
plt.title('Training and Validation loss')
plt.legend

plt.show()

model.save("trvgg16.h5")
```

I. VGG19.ipynb Code

```
import tensorflow as tf
import numpy as np
import gc
import matplotlib.pyplot as plt
import pandas as pd
import time
from PIL import Image
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
from keras.layers import Flatten, Dense ,Dropout
from keras.models import Sequential, Model
from sklearn.model_selection import train_test_split

from keras.preprocessing import image
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint
from keras.applications.xception import preprocess_input



from keras.applications.vgg19 import VGG19
base_model = VGG19(input_shape=(250,750, 3), weights='imagenet', include_top=False)

for layer in base_model.layers[:]:
layer.trainable = False

# Check the trainable status of the individual layers
for layer in base_model.layers:
print(layer, layer.trainable)



x = base_model.output
x = GlobalAveragePooling2D()(x)
```

```
x = Dense(1024, activation='relu')(x)
x = Dense(1024, activation='relu')(x)
x = Dense(512, activation='relu')(x)
x = Dense(512, activation='relu')(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.25)(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.25)(x)
predictions = Dense(2, activation='softmax')(x)

model = Model(base_model.input, predictions)
print(model.summary())

from keras.optimizers import *
from keras.callbacks import ReduceLROnPlateau

import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

base_dir = 'C:/Users/HPC/Desktop/full'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'test')

train_datagen = ImageDataGenerator(
rescale=1./255,
rotation_range=30,
width_shift_range=0.3,
height_shift_range=0.3,
zoom_range=0.3,
horizontal_flip=True,
)

validation_datagen = ImageDataGenerator(rescale=1./255)


train_batchsize = 4
val_batchsize = 4

train_generator = train_datagen.flow_from_directory(
train_dir,
target_size=(250, 750),
batch_size=train_batchsize,
class_mode='categorical'

)

validation_generator = validation_datagen.flow_from_directory(
```

```
validation_dir,
target_size=(250, 750),
batch_size=val_batchsize,
class_mode='categorical',
shuffle=False)
print(train_generator.class_indices)

from keras import optimizers
model.compile(loss='categorical_crossentropy',
optimizer=optimizers.RMSprop(lr=0.0001),
metrics=['acc'])
learning_rate_reduction = ReduceLROnPlateau(monitor='acc',
patience=3,
verbose=1,
factor=0.5,
min_lr=0.00001)
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=2)

history = model.fit_generator(
train_generator,
steps_per_epoch=train_generator.samples/train_generator.batch_size ,
epochs=1000,
validation_data=validation_generator,
validation_steps=validation_generator.samples/validation_generator.batch_size,
class_weight='auto',
verbose=1,
callbacks=[learning_rate_reduction,early_stopping])

import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label="Training acc")
plt.plot(epochs, val_acc, 'b', label='Validation_acc')
plt.title('Training and Validation accuracy')
plt.legend

plt.figure()

plt.plot(epochs, loss, 'bo', label="Training loss")
plt.plot(epochs, val_loss, 'b', label='Validation_loss')
plt.title('Training and Validation loss')
```

```
plt.legend

plt.show()

model.save("trvgg19.h5")
```

III. ResNet50.ipynb Code

```
import tensorflow as tf
import numpy as np
import gc
import matplotlib.pyplot as plt
import pandas as pd
import time
from PIL import Image
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
from keras.layers import Flatten, Dense ,Dropout
from keras.models import Sequential, Model
from sklearn.model_selection import train_test_split

from keras.preprocessing import image
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint
from keras.applications.xception import preprocess_input



from keras.applications.resnet50 import ResNet50
base_model = ResNet50(input_shape=(250,750, 3), weights='imagenet', include_top=False)

for layer in base_model.layers[:]:
layer.trainable = False

# Check the trainable status of the individual layers
for layer in base_model.layers:
print(layer, layer.trainable)



x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
x = Dense(1024, activation='relu')(x)
x = Dense(512, activation='relu')(x)
x = Dense(512, activation='relu')(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.25)(x)
x = Dense(128, activation='relu')(x)
```

```
x = Dropout(0.25)(x)
predictions = Dense(2, activation='softmax')(x)

model = Model(base_model.input, predictions)
print(model.summary())

from keras.optimizers import *
from keras.callbacks import ReduceLROnPlateau

import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

base_dir = 'C:/Users/HPC/Desktop/full'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'test')

train_datagen = ImageDataGenerator(
rescale=1./255,
rotation_range=30,
width_shift_range=0.3,
height_shift_range=0.3,
zoom_range=0.3,
horizontal_flip=True,
)

validation_datagen = ImageDataGenerator(rescale=1./255)


train_batchsize = 4
val_batchsize = 4

train_generator = train_datagen.flow_from_directory(
train_dir,
target_size=(250, 750),
batch_size=train_batchsize,
class_mode='categorical'

)

validation_generator = validation_datagen.flow_from_directory(
validation_dir,
target_size=(250, 750),
batch_size=val_batchsize,
class_mode='categorical',
shuffle=False)
print(train_generator.class_indices)
```

```
from keras import optimizers
model.compile(loss='categorical_crossentropy',
optimizer=optimizers.RMSprop(lr=0.0001),
metrics=['acc'])
learning_rate_reduction = ReduceLROnPlateau(monitor='acc',
patience=3,
verbose=1,
factor=0.5,
min_lr=0.00001)
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=2)

history = model.fit_generator(
train_generator,
steps_per_epoch=train_generator.samples/train_generator.batch_size ,
epochs=1000,
validation_data=validation_generator,
validation_steps=validation_generator.samples/validation_generator.batch_size,
class_weight='auto',
verbose=1,
callbacks=[learning_rate_reduction,early_stopping])

import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label="Training acc")
plt.plot(epochs, val_acc, 'b', label='Validation_acc')
plt.title('Training and Validation accuracy')
plt.legend

plt.figure()

plt.plot(epochs, loss, 'bo', label="Training loss")
plt.plot(epochs, val_loss, 'b', label='Validation_loss')
plt.title('Training and Validation loss')
plt.legend

plt.show()

model.save("trresnet.h5")
```

IV. Xception.ipynb Code

```
import tensorflow as tf
import numpy as np
import gc
import matplotlib.pyplot as plt
import pandas as pd
import time
from PIL import Image
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
from keras.layers import Flatten, Dense ,Dropout
from keras.models import Sequential, Model
from sklearn.model_selection import train_test_split

from keras.preprocessing import image
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint
from keras.applications.xception import preprocess_input


from keras.applications.xception import Xception
base_model = Xception(input_shape=(250,750, 3), weights='imagenet', include_top=False)

for layer in base_model.layers[:]:
layer.trainable = False

# Check the trainable status of the individual layers
for layer in base_model.layers:
print(layer, layer.trainable)


x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
x = Dense(1024, activation='relu')(x)
x = Dense(512, activation='relu')(x)
x = Dense(512, activation='relu')(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.25)(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.25)(x)
predictions = Dense(2, activation='softmax')(x)

model = Model(base_model.input, predictions)
print(model.summary())

from keras.optimizers import *
```

```
from keras.callbacks import ReduceLROnPlateau

import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

base_dir = 'C:/Users/HPC/Desktop/full'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'test')

train_datagen = ImageDataGenerator(
rescale=1./255,
rotation_range=30,
width_shift_range=0.3,
height_shift_range=0.3,
zoom_range=0.3,
horizontal_flip=True,
)

validation_datagen = ImageDataGenerator(rescale=1./255)


train_batchsize = 4
val_batchsize = 4

train_generator = train_datagen.flow_from_directory(
train_dir,
target_size=(250, 750),
batch_size=train_batchsize,
class_mode='categorical'

)

validation_generator = validation_datagen.flow_from_directory(
validation_dir,
target_size=(250, 750),
batch_size=val_batchsize,
class_mode='categorical',
shuffle=False)
print(train_generator.class_indices)

from keras import optimizers
model.compile(loss='categorical_crossentropy',
optimizer=optimizers.RMSprop(lr=0.0001),
metrics=['acc'])
learning_rate_reduction = ReduceLROnPlateau(monitor='acc',
patience=3,
verbose=1,
```

```
factor=0.5,
min_lr=0.00001)
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=2)

history = model.fit_generator(
train_generator,
steps_per_epoch=train_generator.samples/train_generator.batch_size ,
epochs=1000,
validation_data=validation_generator,
validation_steps=validation_generator.samples/validation_generator.batch_size,
class_weight='auto',
verbose=1,
callbacks=[learning_rate_reduction,early_stopping])

import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label="Training acc")
plt.plot(epochs, val_acc, 'b', label='Validation_acc')
plt.title('Training and Validation accuracy')
plt.legend

plt.figure()

plt.plot(epochs, loss, 'bo', label="Training loss")
plt.plot(epochs, val_loss, 'b', label='Validation_loss')
plt.title('Training and Validation loss')
plt.legend

plt.show()

model.save("trxception.h5")
```

Appendix D

Model Fine Tuning Training Code

I. VGG16.ipynb Code

```
import tensorflow as tf
import numpy as np
import gc
import matplotlib.pyplot as plt
import pandas as pd
import time
from PIL import Image
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
from keras.layers import Flatten, Dense ,Dropout
from keras.models import Sequential, Model
from sklearn.model_selection import train_test_split

from keras.preprocessing import image
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint
from keras.applications.xception import preprocess_input



from keras.applications.vgg16 import VGG16
base_model = VGG16(input_shape=(250,750, 3), weights='imagenet', include_top=False)

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
x = Dense(1024, activation='relu')(x)
x = Dense(512, activation='relu')(x)
```

```
x = Dense(512, activation='relu')(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.25)(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.25)(x)
predictions = Dense(2, activation='softmax')(x)

model = Model(base_model.input, predictions)
print(model.summary())

from keras.optimizers import *
from keras.callbacks import ReduceLROnPlateau

import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

base_dir = 'C:/Users/HPC/Desktop/full'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'test')

train_datagen = ImageDataGenerator(
rescale=1./255,
rotation_range=30,
width_shift_range=0.3,
height_shift_range=0.3,
zoom_range=0.3,
horizontal_flip=True,
)

validation_datagen = ImageDataGenerator(rescale=1./255)


train_batchsize = 4
val_batchsize = 4

train_generator = train_datagen.flow_from_directory(
train_dir,
target_size=(250, 750),
batch_size=train_batchsize,
class_mode='categorical'

)

validation_generator = validation_datagen.flow_from_directory(
validation_dir,
target_size=(250, 750),
batch_size=val_batchsize,
```

```
class_mode='categorical',
shuffle=False)
print(train_generator.class_indices)

from keras import optimizers
model.compile(loss='categorical_crossentropy',
optimizer=optimizers.RMSprop(lr=0.0001),
metrics=['acc'])
learning_rate_reduction = ReduceLROnPlateau(monitor='acc',
patience=3,
verbose=1,
factor=0.5,
min_lr=0.00001)
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=2)

history = model.fit_generator(
train_generator,
steps_per_epoch=train_generator.samples/train_generator.batch_size ,
epochs=1000,
validation_data=validation_generator,
validation_steps=validation_generator.samples/validation_generator.batch_size,
class_weight='auto',
verbose=1,
callbacks=[learning_rate_reduction,early_stopping])

import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label="Training acc")
plt.plot(epochs, val_acc, 'b', label='Validation_acc')
plt.title('Training and Validation accuracy')
plt.legend

plt.figure()

plt.plot(epochs, loss, 'bo', label="Training loss")
plt.plot(epochs, val_loss, 'b', label='Validation_loss')
plt.title('Training and Validation loss')
plt.legend

plt.show()
```

```
model.save("vgg16.h5")
```

II. VGG19.ipynb Code

```
import tensorflow as tf
import numpy as np
import gc
import matplotlib.pyplot as plt
import pandas as pd
import time
from PIL import Image
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
from keras.layers import Flatten, Dense ,Dropout
from keras.models import Sequential, Model
from sklearn.model_selection import train_test_split

from keras.preprocessing import image
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint
from keras.applications.xception import preprocess_input


from keras.applications.vgg19 import VGG19
base_model = VGG19(input_shape=(250,750, 3), weights='imagenet', include_top=False)

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
x = Dense(1024, activation='relu')(x)
x = Dense(512, activation='relu')(x)
x = Dense(512, activation='relu')(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.25)(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.25)(x)
predictions = Dense(2, activation='softmax')(x)

model = Model(base_model.input, predictions)
print(model.summary())

from keras.optimizers import *
from keras.callbacks import ReduceLROnPlateau

import os
import numpy as np
```

```
from keras.preprocessing.image import ImageDataGenerator

base_dir = 'C:/Users/HPC/Desktop/full'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'test')

train_datagen = ImageDataGenerator(
rescale=1./255,
rotation_range=30,
width_shift_range=0.3,
height_shift_range=0.3,
zoom_range=0.3,
horizontal_flip=True,
)

validation_datagen = ImageDataGenerator(rescale=1./255)


train_batchsize = 4
val_batchsize = 4

train_generator = train_datagen.flow_from_directory(
train_dir,
target_size=(250, 750),
batch_size=train_batchsize,
class_mode='categorical'

)

validation_generator = validation_datagen.flow_from_directory(
validation_dir,
target_size=(250, 750),
batch_size=val_batchsize,
class_mode='categorical',
shuffle=False)
print(train_generator.class_indices)

from keras import optimizers
model.compile(loss='categorical_crossentropy',
optimizer=optimizers.RMSprop(lr=0.0001),
metrics=['acc'])
learning_rate_reduction = ReduceLROnPlateau(monitor='acc',
patience=3,
verbose=1,
factor=0.5,
min_lr=0.00001)
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=2)
```

```
history = model.fit_generator(
train_generator,
steps_per_epoch=train_generator.samples/train_generator.batch_size ,
epochs=1000,
validation_data=validation_generator,
validation_steps=validation_generator.samples/validation_generator.batch_size,
class_weight='auto',
verbose=1,
callbacks=[learning_rate_reduction,early_stopping])

import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label="Training acc")
plt.plot(epochs, val_acc, 'b', label='Validation_acc')
plt.title('Training and Validation accuracy')
plt.legend

plt.figure()

plt.plot(epochs, loss, 'bo', label="Training loss")
plt.plot(epochs, val_loss, 'b', label='Validation_loss')
plt.title('Training and Validation loss')
plt.legend

plt.show()

model.save("vgg19.h5")
```

III. ResNet50.ipynb Code

```
import tensorflow as tf
import numpy as np
import gc
import matplotlib.pyplot as plt
import pandas as pd
import time
from PIL import Image
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
```

```
from keras.layers import Flatten, Dense ,Dropout
from keras.models import Sequential, Model
from sklearn.model_selection import train_test_split

from keras.preprocessing import image
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint
from keras.applications.xception import preprocess_input



from keras.applications.resnet50 import ResNet50
base_model = ResNet50(input_shape=(250,750, 3), weights='imagenet', include_top=False)

x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
x = Dense(1024, activation='relu')(x)
x = Dense(512, activation='relu')(x)
x = Dense(512, activation='relu')(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.25)(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.25)(x)
predictions = Dense(2, activation='softmax')(x)

model = Model(base_model.input, predictions)
print(model.summary())

from keras.optimizers import *
from keras.callbacks import ReduceLROnPlateau

import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

base_dir = 'C:/Users/HPC/Desktop/full'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'test')

train_datagen = ImageDataGenerator(
rescale=1./255,
rotation_range=30,
width_shift_range=0.3,
height_shift_range=0.3,
zoom_range=0.3,
horizontal_flip=True,
)
```

```
validation_datagen = ImageDataGenerator(rescale=1./255)



train_batchsize = 4
val_batchsize = 4

train_generator = train_datagen.flow_from_directory(
train_dir,
target_size=(250, 750),
batch_size=train_batchsize,
class_mode='categorical'

)

validation_generator = validation_datagen.flow_from_directory(
validation_dir,
target_size=(250, 750),
batch_size=val_batchsize,
class_mode='categorical',
shuffle=False)
print(train_generator.class_indices)

from keras import optimizers
model.compile(loss='categorical_crossentropy',
optimizer=optimizers.RMSprop(lr=0.0001),
metrics=['acc'])
learning_rate_reduction = ReduceLROnPlateau(monitor='acc',
patience=3,
verbose=1,
factor=0.5,
min_lr=0.00001)
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=2)

history = model.fit_generator(
train_generator,
steps_per_epoch=train_generator.samples/train_generator.batch_size ,
epochs=1000,
validation_data=validation_generator,
validation_steps=validation_generator.samples/validation_generator.batch_size,
class_weight='auto',
verbose=1,
callbacks=[learning_rate_reduction,early_stopping])

import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
```

```
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label="Training acc")
plt.plot(epochs, val_acc, 'b', label='Validation_acc')
plt.title('Training and Validation accuracy')
plt.legend

plt.figure()

plt.plot(epochs, loss, 'bo', label="Training loss")
plt.plot(epochs, val_loss, 'b', label='Validation_loss')
plt.title('Training and Validation loss')
plt.legend

plt.show()

model.save("resnet50.h5")
```

IV. Xception.ipynb Code

```
import tensorflow as tf
import numpy as np
import gc
import matplotlib.pyplot as plt
import pandas as pd
import time
from PIL import Image
from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D
from keras.layers import Flatten, Dense ,Dropout
from keras.models import Sequential, Model
from sklearn.model_selection import train_test_split

from keras.preprocessing import image
from keras.callbacks import EarlyStopping
from keras.callbacks import ModelCheckpoint
from keras.applications.xception import preprocess_input
from keras.applications.vgg16 import VGG16
from keras.applications.xception import Xception

base_model = Xception(input_shape=(250,750, 3), weights='imagenet', include_top=False)

x = base_model.output
x = GlobalAveragePooling2D()(x)
```

```
x = Dense(1024, activation='relu')(x)
x = Dense(1024, activation='relu')(x)
x = Dense(512, activation='relu')(x)
x = Dense(512, activation='relu')(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.25)(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.25)(x)
predictions = Dense(2, activation='softmax')(x)

model = Model(base_model.input, predictions)
print(model.summary())

from keras.optimizers import *
from keras.callbacks import ReduceLROnPlateau

import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

base_dir = 'C:/Users/HPC/Desktop/full'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'test')

train_datagen = ImageDataGenerator(
rescale=1./255,
rotation_range=30,
width_shift_range=0.3,
height_shift_range=0.3,
zoom_range=0.3,
horizontal_flip=True,
)

validation_datagen = ImageDataGenerator(rescale=1./255)


train_batchsize = 4
val_batchsize = 4

train_generator = train_datagen.flow_from_directory(
train_dir,
target_size=(250, 750),
batch_size=train_batchsize,
class_mode='categorical'

)

validation_generator = validation_datagen.flow_from_directory(
```

```
validation_dir,
target_size=(250, 750),
batch_size=val_batchsize,
class_mode='categorical',
shuffle=False)
print(train_generator.class_indices)

from keras import optimizers
model.compile(loss='categorical_crossentropy',
optimizer=optimizers.RMSprop(lr=0.0001),
metrics=['acc'])
learning_rate_reduction = ReduceLROnPlateau(monitor='acc',
patience=3,
verbose=1,
factor=0.5,
min_lr=0.00001)
from keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=10, verbose=2)

history = model.fit_generator(
train_generator,
steps_per_epoch=train_generator.samples/train_generator.batch_size ,
epochs=1000,
validation_data=validation_generator,
validation_steps=validation_generator.samples/validation_generator.batch_size,
class_weight='auto',
verbose=1,
callbacks=[learning_rate_reduction,early_stopping])

import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc) + 1)

plt.plot(epochs, acc, 'bo', label="Training acc")
plt.plot(epochs, val_acc, 'b', label='Validation_acc')
plt.title('Training and Validation accuracy')
plt.legend

plt.figure()

plt.plot(epochs, loss, 'bo', label="Training loss")
plt.plot(epochs, val_loss, 'b', label='Validation_loss')
plt.title('Training and Validation loss')
```

```
plt.legend

plt.show()

model.save("trxce.h5")
```

Appendix E

Verify True Accuracy Code

I. VerifyTrueAccuracy.ipynb Code

```
import os
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import cv2
import numpy as np
from glob import glob as gb
import matplotlib.image as mpimg
import seaborn as sns
get_ipython().run_line_magic('matplotlib', 'inline')
np.random.seed(2)
from keras.applications.xception import Xception
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import itertools

from keras.utils.np_utils import to_categorical
from keras.optimizers import RMSprop
from keras.callbacks import ReduceLROnPlateau
import sys
from keras.layers import *
from keras.optimizers import *
from keras.applications import *
from keras.models import Model
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras import backend as k
```

```
def convert2label(vector):
string_array=[]
for results in vector:
if results==0:
string_array.append('low')
elif results==1:
string_array.append('upp')
return string_array

test = []
img  = cv2.imread('./true/001_20.jpg')
plt.figure(figsize=(5,5))
img2 = cv2.resize(img, (750,250), interpolation=cv2.INTER_CUBIC)

imga = np.asarray(img2)
test.append(imga)
test = np.array(test)
test.shape


test = test.astype('float32') / 255.0

from keras.models import load_model

model = load_model("./trvgg160525.h5")

ss = model.predict(test)

a=np.array(ss).round(decimals=5)
print(a)
results = np.argmax(ss,axis = 1)
y = convert2label(results)

plt.figure(figsize=(5,5))
plt.imshow(img)

if a[0,0]<0.4 and a[0,1]<0.4 and a[0,2]<0.4 and a[0,3]<0.4 and a[0,4]<0.4:
print('error')
else :
print(np.max(a))
print(results)
print(y)
```