四川大學

SICHUAN UNIVERSITY

本科生毕业设计（学术论文）
Undergraduate Graduation Project (Academic Thesis)

题　　目	Convolutional Neural Network for Skin Cancer Detection and Classification
学　　院	软件学院
专　　业	软件工程专业
学生姓名	Md Naim Hassan Saykat
学　　号	2018511460042　年级 2018 (Spring)

Title **Convolutional Neural Network for Skin Cancer Detection and Classification**

School **College of Software Engineering**

Major **Software Engineering**

Student's Name **Md Naim Hassan Saykat**

Student ID: **2018511460042** Grade **2018 (Spring)**

Adviser **Chuan Li**

教务处制表

二O一八年五月二十日

Made by the Office of Academic Affairs

May 20, 2018

ABSTRACT

Software Engineering

Student: Md Naim Hassan Saykat Adviser: Chuan Li

The skin cancer is a cancerous mass or growth of abnormal cells on the skin. The abnormal growth of cells causes skin lesions. Skin lesions can be melanocytic nevi, melanoma, benign keratosis-like lesion, basal cell carcinoma, actinic keratoses, vascular lesion or dermatofibroma. The WHO predicted that between 2 and 3 million non-melanoma skin cancers and 132,000 melanoma skin cancers appear globally each year. Recent progression in deep learning has helped the health industry in medical imaging for medical diagnostic of numerous diseases. For Visual learning and Image Recognition, task Convolutional Neural Network (CNN) is the most popular and generally used machine learning algorithm. This study presents a new CNN architecture for skin cancer classification into seven classes from dermoscopic images. The experimental result indicates that the proposed CNN model accuracy result is very effective and has a meager complexity rate on achieving 81% validation accuracy. In comparison, VGG-16 achieved 76%, Xception achieved 73%, and ResNet-50 achieved 79% accuracy on test set. The proposed CNN model requires significantly less computational power and has much better accuracy than other pre-trained models.

[Key Words] Deep Learning, CNN, Transfer Learning, Xception, VGG-16, ResNet-50

Table of Contents

1. Preamble **5**
1.1 Introduction 5
1.2 Background 5
1.4 Organization and Structure 7
2. Introduction to the Basic Algorithm or Background Knowledge **8**
2.1 Introduction to CNN, Loss Function and Optimization 8
2.2 Transfer Learning 13
2.3 Dataset 17
3 Methodology, Implementation and Evaluation **18**
3.1 Loading Metadata 18
3.2 Loading and Preprocessing Images 19
3.3 Model Architecture 19
3.4 Training the model 21
3.5 Evaluating the model 22
3.6 VGG-16 training and evaluation 22
3.7 Xception training and evaluation 23
3.8 ResNet-50 training and evaluation 24
3.9 Performance Comparison 25
4. Web Application **28**
4.1 Development 28
4.2 Backend 28
4.3 Frontend 28
5. Conclusion and Future Work **30**
5.1 Conclusion 30
5.2 Future Work 30
References **31**
Statement **33**
Acknowledgement **34**

1. Preamble

1.1 Introduction

Skin is the most significant and most sensitive part of the human body, which saves our internal vital regions and organs from the outside environment, avoiding connection with bacteria and viruses. Skin also assists in body temperature regulation. The skin consists of cells, blood vessels, pigmentation, and other components.

Skin Cancer is one of the numerous deathful cancers. The detection of skin cancer in the early stages is an expensive and challenging process. It is bound to spread to different body parts if it is not diagnosed and dined at the commencement time. It is primarily because of the abnormal growth of skin cells, which is often produced when the body is revealed to sunlight. If it is not detected at beginning stage, it is quickly invaded nearby tissues and spread to other parts of the body. Formal diagnosis method to skin cancer detection is Biopsy method. A biopsy is a method to remove a piece of tissue or a sample of cells from patient body so that it can be analysed in a laboratory. It is uncomfortable method. Biopsy Method is time consuming for patient as well as doctor because it takes lot of time for testing. Considering the limited availability of resources, early detection of skin cancer is critical. Proper diagnosis and detection feasibility are vital for skin cancer prevention policy. Skin cancer detection in the early phases is a challenge for even dermatologists. Nevertheless, advances in machine learning have allowed us to detect and classify such cancerous cells, especially by utilizing convolutional neural networks.

One of the noninvasive skin imaging techniques known as dermoscopy is the key to the diagnosis of skin cancer. Dermoscopy exaggerates the region of attraction (ROI) optically and then carries digital images of the ROI. Loss in diagnosis is the leading cause of skin cancer fatalities [1]. These errors are sometimes because of the complexity of the subsurface configurations and the subjectivity of visual variations [2, 3]. Later, automated image detection or classification instruments need to be required to help doctors, or medical support aids in decreasing diagnostic errors. Talented physicians seek the existence of exclusive optical components to analyze skin lesions accurately in most clinical dermoscopy techniques. These exist the characteristics considered for irregularities and malignancy [4, 5, 6, 7]. Nevertheless, in the possibility of an unprofessional dermatologist, diagnosis of skin cancer is usually highly difficult. The precision of skin cancer detection with dermoscopy constantly changes from 75-85% [8]. This demonstrates the demand for better computer-aided diagnosis platforms.

1.2 Background

Among all the organs in human body, skin is the largest. The importance of skin comes from the way it protects the internal body tissues from the external environment, i.e., it keeps the body temperature at a level, protects our body from radiations coming from the sun such as ultraviolet (UV) light exposure, prevents infections, and assists in the synthesis of vitamin D, which is elemental for many body functions [9]. Recently, a significant increase in skin cancer patients has been recorded, and studies publicize that the rate of melanoma patients doubles in every 20 years [10]. Skin cancer one of the principal type of cancer, it is produced when skin cells begin to grow out of control. Skin cancers are mainly of three types: (i) basal cell skin cancers, (ii) squamous cell skin cancers, and (iii) melanomas. Normally, skin cancers which are not melanomas are termed as non-melanoma skin cancers. [11] A deadly form of skin cancer is melanoma which is often misdiagnosed as a benign lesion [12]. There are an approximately 76,380 new cases of melanoma and an approximate 6,750 deaths each year in the US [13]. They also estimated the death rate is higher in male than in male than female. A specialist do visual analysis by analyzing the pigmented leisons by changing size, irregular shape and colour. Then histopathologic diagnosis is another way to examine the cancer cell. Melanoma can be detected by simple visual experiment since it occurs on the skin surface, but early detection and classification is a matter of life and death as the lives of skin cancer patients depend on accurate and early diagnosis. This constitutes additional challenges to the task of distinguishing among skin lesions, especially between benign or malignant tumors, due to the significant imbalance in the number of samples of each classes of lesions. These facts must be looked at carefully when designing skin lesions classifier systems whose performance should be compatible with traditional detection methods. That is the reason for distinguishing among skin lesions, especially between benign or malignant cancer, which has allowed the advent of automatic skin lesions classification to compete against traditional detection methods. Most current approaches in the field of skin lesion classification rely on hand-crafted features, such as the ABCDE rule (Asymmetry, Border, Color, Dermoscopic structure, and Evolving) [14], 7-point checklist [16], Menzies method [17] 3-point checklist [15], and Color, Architecture, Symmetry, and Homogeneity (CASH). Physicians often depend on personal understanding and assess each patient's lesions by taking into account the patient's local lesion imprints in comparison to that of the entire body. Without any computer-based assistance, the clinical diagnosis accuracy for melanoma detection is reported to be between 65 and 80%. The use of dermoscopic images, and pictures taken by skin surface microscopy, improves the diagnostic accuracy of skin lesions by 49%. However, even for trained medical experts the visual differences between melanoma and benign skin lesions can be very tenuous, making it difficult to classify the different classes. For the given reasons, an intelligent medical imaging-based skin lesion diagnosis system can be a beneficial tool to help a physician in classification of skin lesions.

1.3 Main Work of This Project

The principal work of this project is to propose a system that detects skin cancer and classifies it in different classes by using the Convolution Neural Network and to analyze the result to see how the model can be useful in practical scenario.

1.4 Organization and Structure

The body of this thesis is organized of five parts, and the part of each component is as follows:

Part I: Introduction. This chapter presents the research background of the project and defines current studies on this project. Eventually, the main work of this project is delivered.

Part II: technological knowledge and technologies applied in this project give a short introduction to convolutional neural networks, deep learning, and other algorithms.

Part III: provides the complete details of the project, including model building, workflow, training, testing, and conclusive results.

Part IV: web application development for dermatologists.

Part V: This chapter gives a summary of the whole project.

2. Introduction to the Basic Algorithm or Background Knowledge

2.1 Introduction to CNN, Loss Function and Optimization

Convolutional Neural Network (CNN) is a Deep Learning algorithm. It can receive an input image, assign importance weights and biases to various aspects/objects in the picture, and distinguish one from another. The pre-processing needed in a CNN model is much lower as compared to other classification algorithms. In the primitive methods, filters are hand-engineered. With enough training, CNN can learn these filters/characteristics. The architecture of a CNN is similar to the connectivity pattern of neurons in the human brain.

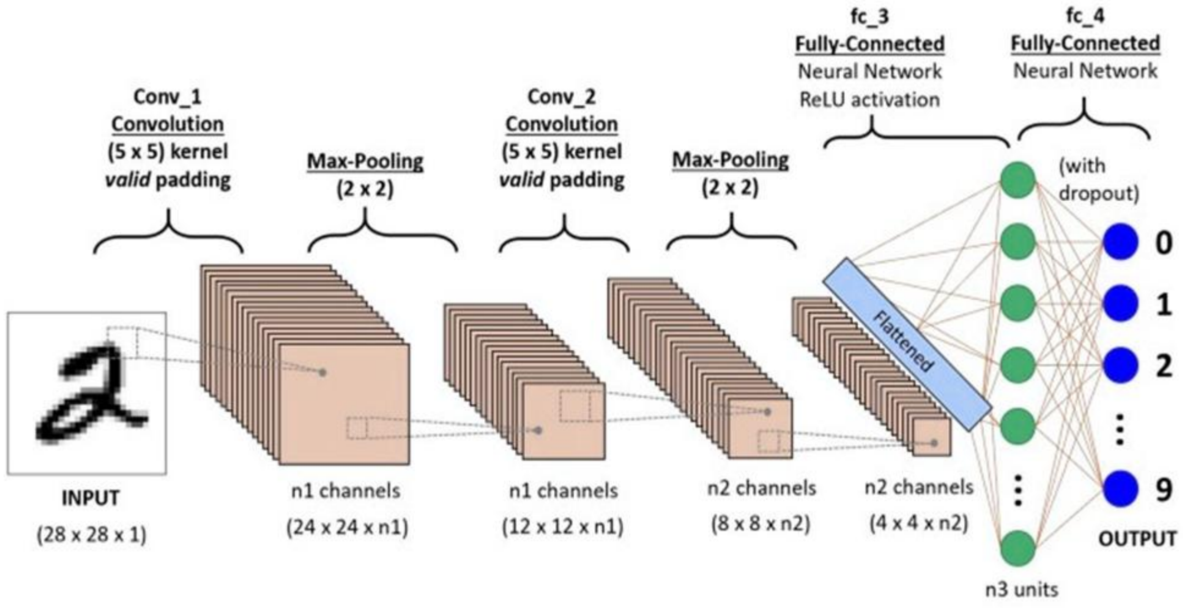


Figure 1: An Example of CNN architecture

A CNN can capture the spatial and temporal dependencies in an image through the application of relevant filters. The CNN architecture performs a better flitting to the image dataset because of the reduced number of parameters involved and the usability of weights. In other words, the network can train to understand the complexity of the image better. The convolution operation aims to extract high-level features similar to edges from the input image. Conventionally, the first CNN layer is liable for capturing the Low-level features such as edges, color, gradient exposure, Etc. By adding layers, the Architecture adapts to the high-position features, giving us a network with a better understanding of images in the dataset, similar to how the human brain does. There are two types of results in the function: First, the convolved point is reduced in dimensionality compared to the input, and second, the dimensionality is increased or stays the same. This is done by using Valid Padding in the case of the former or the Same Padding in the case of the latter.

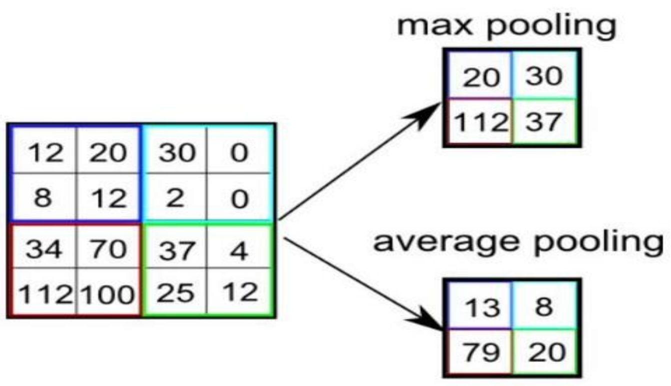


Figure 2: Max Pooling

Duplicate as the convolutional layer. The pooling layer is also exposed to reduce the spatial measure of the convolved characteristic. The pooling layer reduces the computational ability to reprocess the data via dimensionality decrease. Similarly, it supports extracting predominant features, which are positional consistent and rotational, thus supporting functional training of the model. There are two kinds of pooling: Max Pooling and Average Pooling. First of all, Max Pooling replaces the highest value from the amount of the image surrounded by the kernel. Secondly, Average Pooling delivers the average of all the deals from the image part covered by the kernel. Max Pooling also operates as a Noise Suppressant. It tosses out the loud activation and also serves de-noising along with the dimensionality lowering.

On the other hand, Average Pooling decreases dimensionality like a noise hiding process. Therefore, we can express that Max Pooling achieves more suitability than Average Pooling. The Pooling Layer and the Convolutional Layer form the i-th layer of a CNN algorithm. Depending on the difficulties in the images, the number of matching layers may be enriched for developing low-level elements even distant, but at the expense of more computational ability. The final result is flattened and provided to a standard Neural Network for classification ambitions. Adding a Fully-connected layer is a standard way of discovering non-linear mixtures of the high-position elements as defined by the experience of the convolutional layer. The Fully- Combined layer understands a maybe non-linear operation in that space.

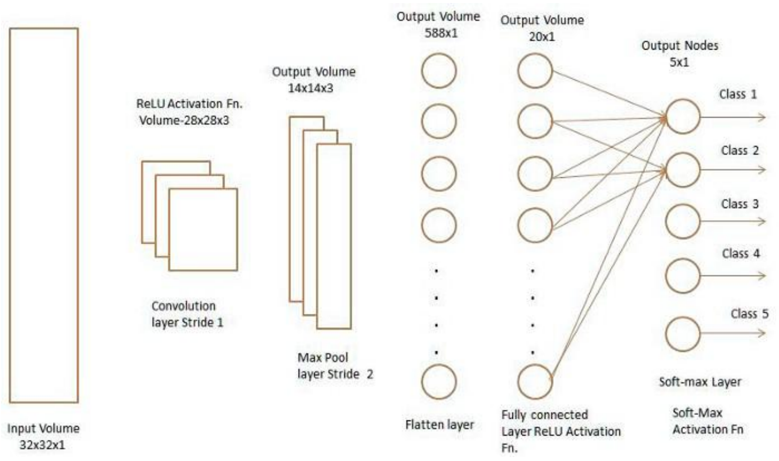


Figure 3: CNN classification-fully connected layer CNN classification layer description

Now that the input image has converted into a suitable form, it also flattened it into a column vector. The flattened output is fed to a forward-propagation neural network, and back-propagation is applied to the training of every iteration. After a few epochs, the model is suitable for distinguishing between skin cancers and certain low-level features in images and classifying them using the softmax classification technique.

Batch Normalization: Batch normalization solves a significant problem called internal co-variate shift. It helps to make the data flowing between intermediate layers of the neural network look, which means a higher learning rate can be used. It has a formalizing effect which means constantly can remove dropouts. Batch normalization is an approach for training genuinely deep neural networks that standardizes the inputs to a layer for each mini-batch. It stabilizes the learning process and dramatically reduces the number of training epochs needed to train deep networks.

Activation Function: The proposed CNN model has been built with two activation functions:

(1) ReLU: The Rectified Linear Unit (ReLU) activation function per convolutional layer has been used. An Activation function transforms the weighted input sum into that node's result, as illustrated by Vinod and Hinton [18]. Rectifier Linear unit function is often utilized in secret convolutional neural network layers. Mathematically, ReLU is defined by

$$f(z) = \max(0, x) \tag{1}$$

Where z is the input when z is negative or equivalent to 0, it transforms the negative input to 0. When the input is greater than 0, the output will be 1. So, the result of ReLUs will be

$$f'(x) = \begin{cases} 1, & \text{for} x >= 0 \\ 0, & \text{for} x < 0 \end{cases} \tag{2}$$

So if the input is 0, that neuron is a dead neuron in ReLU function, and it won't be activated.

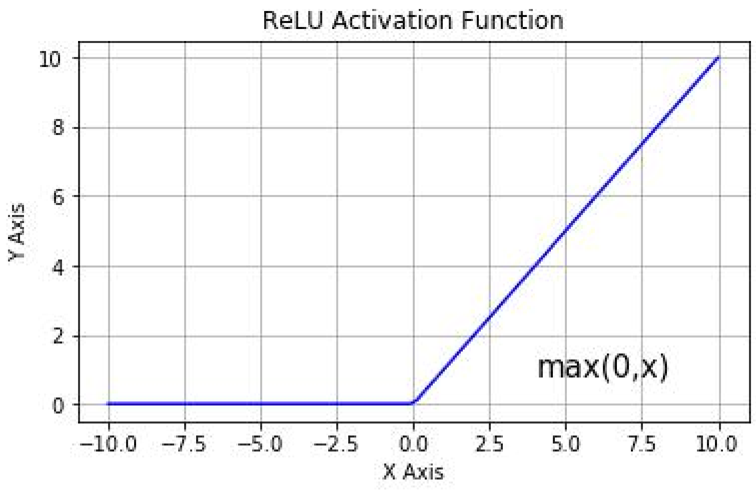


Figure 4: ReLU activation function

(2) Softmax: The softmax function is the activation function in the output layer of neural network models that predict a

multinomial probability distribution. Softmax is usually used to activate multi-class classification problems requiring class membership on more than two class labels.

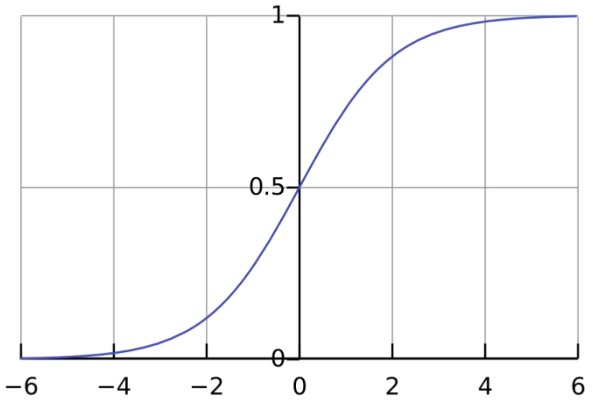


Figure 5: Softmax activation function

Loss Function: In machine learning, people utilize the loss function to sum the error between the actual label values and algorithmic predictions. It is the default loss function for the multi-class classification problems where every class is appointed a particular integer value from 0 to (num_classes – 1). It will compute the average difference between the actual and predicted probability distributions for all classes in the problem. The score is reduced and a perfect cross-entropy value is 0. In this experiment, the Categorical Cross-Entropy loss function has been used. The categorical cross-entropy loss function calculates the loss by computing the following sum:

$$\text{Loss} = -\sum_{i=1}^{\text{output}} y_i \cdot \log \hat{y}_i \tag{3}$$

Where ŷi is the i-th scalar value in the model output, yi is the corresponding target value, and the output size is the number of scalar values in the model output. This loss is a perfect measure of how distinguishable two discrete probability distributions are from each other. In this context, yi is the probability that event i occurs, and the sum of all yi is 1, meaning that precisely one event may occur. The minus sign confirms that the loss is reduced when the distributions get closer to each other. The target needs to be one-hot encoded. This presents them directly accurate to use with the categorical cross-entropy loss function. The output layer is computed with n nodes (one for each class), in this MNIST case, 10 nodes, and a "softmax" activation has used in order to predict the probability for each class.

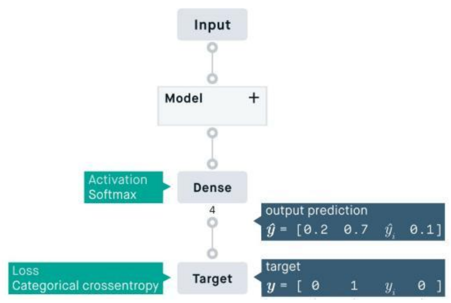


Figure 6: Categorical Cross-Entropy

Optimizers:

1. Stochastic Gradient Descent: The Stochastic Gradient Descent technique is presented by Herbert and Sutton [19]. In short Stochastic Gradient Descent, we accept the result of weights, dW, and derivative of Bias, DB, for each epoch. Moreover, multiply with the learning rate.

$$W = W - \eta \times d \tag{4}$$

$$b = b - \eta \times db \tag{5}$$

While Stochastic Gradient Descent with momentum V is the moving mean of our gradients, here is the moving mean between 0 and 1 when we calculate dW and db on the current batch.

$$V_{dW} = \beta \times V_{dW} + (1 - \beta) \times d \tag{6}$$

$$V_{db} = \beta \times V_{db} + (1 - \beta) \times db \tag{7}$$

To compile the proposed CNN model, categorical cross-entropy has been used because, in this case, it is a multi-class classification task. Formally, it is designed to determine the difference between two probability distributions.[20]
It can reduce the computation cost of gradient while still largely maintaining the gradient direction when averaged over many mini-batches or samples. The SGD optimizer has been used to compile the proposed CNN model.

2. RMSprop: The RMSprop optimizer restricts the changes in the perpendicular direction. Thus, increasing the learning rate and proposed CNN algorithm could take more significant steps in the vertical direction clustering faster. The difference between RMSprop and gradient descent is in how the gradients are calculated. As with SGD, Root Mean Squared Prop is an adaptive learning rate procedure proposed by Geoff and Hinton [21]. In RMSprop, the exponential satisfying mean square of gradients is used. So, in RMSprop,

$$S_{dW} = \beta \times S_{dW} + (1 - \beta) \times dW^2 \tag{8}$$

$$S_{db} = \beta \times S_{db} + (1 - \beta) \times d \tag{9}$$

Beta β is a hyperparameter that handles exponentially weighted means.

RMSprop optimizer has been utilized to collect the Inception-V3 Model.

3. Adam: Adam is the best among the adaptive optimizers in most cases. Good with sparse data: the adaptive learning rate is perfect for this type of dataset. Building upon the strengths of transfer learning models, Adam optimizer gives much higher performance than the previously used. It outperforms them by a significant margin into giving an optimized gradient descent. So incorporating the characteristics of the weighted mean of past gradients and the weighted mean of the squares of the past gradients, we execute the Adam optimization technique, So the updated weights and bias in the Adam optimizer will be

$$W = W - \eta \times \left(V_{dw} / \sqrt{S_{dw} + \epsilon}\right) \tag{10}$$

$$b = b - \eta \times \left(V_{db} / \sqrt{S_{db} + \epsilon}\right) \tag{11}$$

Epsilon ϵ is a small number containing zero division (Epsilon = 0.00000001), and η is a learning rate with various values.

An optimizer trains a model faster and good for developing state-of-the-art deep learning models on a wide variety of popular tasks in the field of CV, NLP, Etc. [22]. Adam optimizer has been used to train the Resnet50 and Inception-V3 model and RMSprop for Xception and VGG-16.

2.2 Transfer Learning

A central theory in many machines learning and data mining algorithms is that the training and future data must have the same feature range and pattern. However, in many applications, this theory will not exist. For example, we sometimes have a classification task in one domain of interest. Still, we only include A central theory in multiple machine learning and data mining algorithms: the training and future data must have the same feature pattern and range. However, in many applications, this theory will not exist. For instance, we occasionally have a classification task in one field of interest. Still, we only have good training data in another part of the case, where the latter data may be in another part area or observe another data allocation. In such cases, if done successfully, learning transfer would significantly improve learning implementation by avoiding costly data labeling efforts. In recent years, transfer learning has appeared as a new learning framework to identify this problem.

Transfer learning allows neural networks to use significantly less data. We are transferring the knowledge that a model has learned from a previous task to our current one with transfer learning. The idea is that the two studies are not disjoint; as such, we can leverage whatever network parameters that model has learned through its extensive training without doing that ourselves. Transfer learning has been proven to boost model accuracy and reduce required training time, less data, and more accuracy.

Transfer learning is classified into inductive transfer learning, transductive transfer learning, and unsupervised transfer learning. Most previous works focused on the settings. Furthermore, each method to transfer learning can be classified into four contexts based on ―what to transfer in learning. They include the instance-transfer method, the feature-representation-transfer approach, the parameter transfer approach, and the relational knowledge-transfer approach. The smaller networks converged & were then used as initialization for the more extensive, deeper networks- This process is called pre-training. While making logical sense, pre-training is a very time-consuming, slow task requiring an entire network to be trained before initializing a deeper network.

In computer vision, for example, some feature extractors from a bareness detection model can be used to speed up the learning method for the skin cancer classification model.

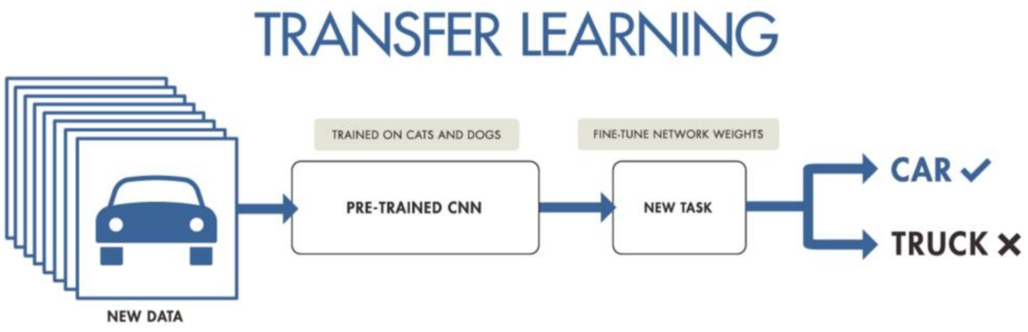


Figure 7: An example of Transfer Learning

Using a pre-trained model can significantly reduce the time needed for feature engineering and training.

Select Source Model: The first step is to take a source model, ideally, one with a large dataset to train. Numerous research institutions release these models and datasets as open-sourced projects, so it is unnecessary to create.

Exercise the model: The coming step is to decide which layers to reuse in the network. The goal is to produce an architecture that is better than a naïve model so that it can be assured that some new feature learning takes place.

Tune the model: Generally, deeper layers are reused as these tend to be more general, whereas the top layers tend to be more finely tuned to a particular problem.

Eventually, the new model is trained on the dataset. The significant advantage is that the model tends to confluence much quick, and therefore more minor data and compute time are needed.

I have used transfer learning to save time or to get better performance as it is an optimization.

There are three possible advantages to look for when using transfer learning:

(1) Higher start:

The original skill on the source model is advanced than it else would be.

(2) Higher slope:

The improvement rate of skill during training of the source model. That is steeper than it else would be.

(3) Higher asymptote:

The gathered skill of the trained model is better than it else would be.

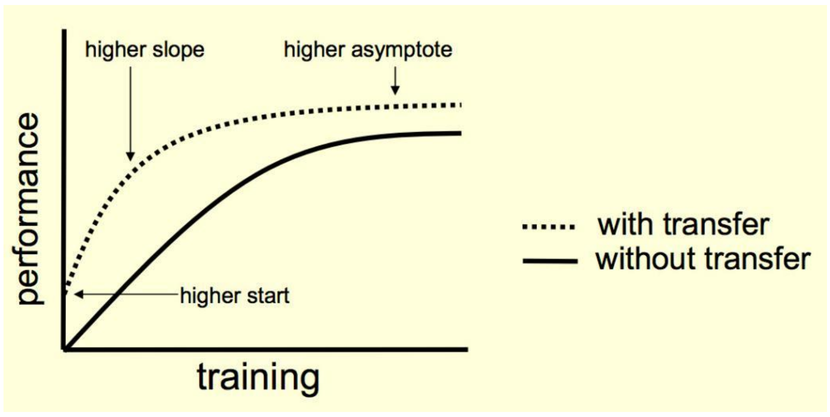


Figure 8: Transfer Learning for Deep Learning With CNN

2.2.1 VGG-16 Model

CNN is a deep learning algorithm that grasps an input image, shows characteristics to different image features, and classifies that image. The pre-processing needed in the Convolution network is much more moderate compared with other classification models. VGG-16 is a convolution neural network 16 layers deep, making the model larger than can enhance the training time, i.e., the input images are processed into 16 layers. In the VGG-16 model, the output is classified into different categories. For example, assume a VGG-16 network takes input images of 100 other objects. The model then classifies the images based on the requirements such as the appearance or absence of a tumor, classifying objects in images. K Simonyan and A Zisserman proposed the VGG-16 model, and they were able to achieve 92.7% accuracy at the top 5 test accuracy in ImageNet Database [23]. For the skin cancer classification project, the image set of dermoscopic scans is taken into consideration, and it classifies different images based on whether the tumor is present or not.

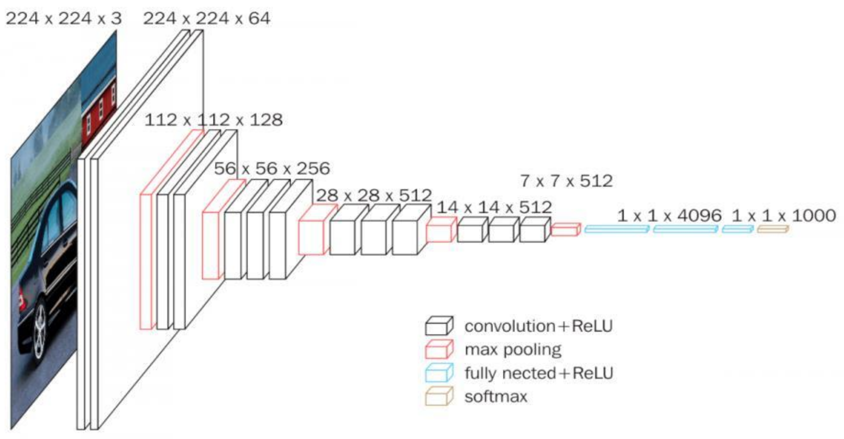


Figure 9: VGG-16 Model

Layers and steps Involved in VGG-16:

VGG_16 model has 16 layers. In every layer, the input image is processed through many steps. A piece of detailed information about the steps is given below,

1) Input the image: Firstly, 224 by 224 pixels input image of the size given to the model.

2) Convolution: Every layer has a 3x3 kernel. In this step, the kernel is convolved with a stride of pixels 2 across the input image, and the feature map is gained. The feature map is the dot product of the input pixel values and the filter. The feature map achieved will pass to the next layer as an input.

3) Max pooling: In this step, the pixel density increases, decreasing the dimension of the feature map. Max pooling uses the maximum value from a 3 x 3 filter with a stride of 2 pixels. The core features are extracted through the max-pooling layer.

4) Normalization: The negative pixel values will convert to 0's. This is done by taking the maximum(xij,0) Rectified Linear Unit (ReLU) function. Through this step, all the unnecessary pixels will be disabled.

5) Dropout: The dropout layer randomly chosen layers are labeled as zeroes so that the network learns new pathways to symbolize the final image. This helps to improve the learning. The softmax function is usually used in the very last step in the neural network. Softmax transforms the output generated in the previous layer into a probability distribution function. This is done by features extracted in the middle layers of the VGG-16 model. The predictions are made based on the results generated by this function.

2.2.2 Xception Model

Francois Chollet proposed the Xception Model, which has been used in image detection [24]. The Xception model is an addition to the inception model. It displaces the standard inception modules with depth-wise separable convolutions. Xception has been trained on the ImageNet database with more than one million images and has the advantage of learning the general features of various photos. The Xception model is fine-tuned on the MNIST-HAM 10000 dataset to classify skin cancer into seven different classes. The parts are extracted from the last layer that is fully connected and provided to the softmax classifier.

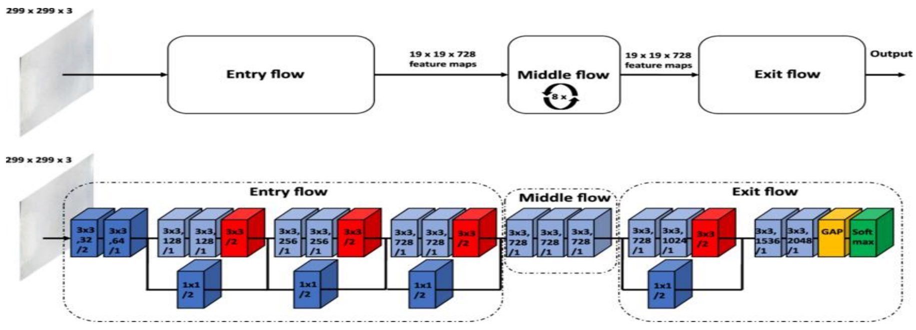


Figure 10: Xception Model

2.2.3 ResNet50 Model

ResNet50 is a 50-layer Residual Network with 26M parameters. Microsoft presented the residual network as a deep convolutional neural network model in 2015 [25]. In a Residual network, we understand residuals removed from aspects learned from the layer's inputs rather than teaching components. ResNet utilized the skip relation to develop information over layers. ResNet attaches nth layer input straight to some (n+x)th layer, allowing further layers to be piled and found in a deep network.

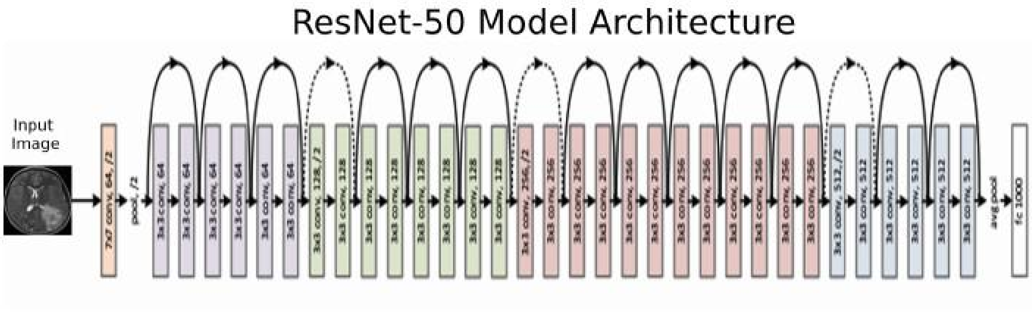


Figure 11: ResNet50 Model

2.3 Dataset

MNIST HAM-10000 dataset for Skin Cancer which is available on Kaggle was used in this study. It consists of 10,015 images of skin pigments and 7 features. The dermatoscopic images are divided amongst seven classes. The numeral of pictures present in the dataset is enough to be used for different tasks including image retrieval, segmentation, feature extraction, deep learning, and transfer learning, etc. The various features of this dataset are:

1. Lesion id

2. Image id

3. Dx: This column includes the types of cancers. Our dataset focuses on 7 major types of skin cancers: a. Melanocytic nevi (nv) b.Melanoma (mel) c. Benign keratosis (bkl) d.Basal cell carcinoma (bcc) e. Actinic keratoses (akiec) f. Vascular lesions (vasc) g. Dermatofibroma (df)

4. Dx_type: This column basically tells us how the cancer was validated. It includes: a.Histopathologic (hist) b. Follow-up c. Confocal d. Consensus:

5. Age: This column basically tells us the age of the patients.

6. Sex: Tells us whether the patient was male or female.

7. Localization: It is basically the part of the body where the cancer was present like back, lower extremity, trunk, upper extremity, abdomen, face, chest, foot, unknown, neck, scalp, hand, ear, genital and acral.

3 Methodology, Implementation and Evaluation

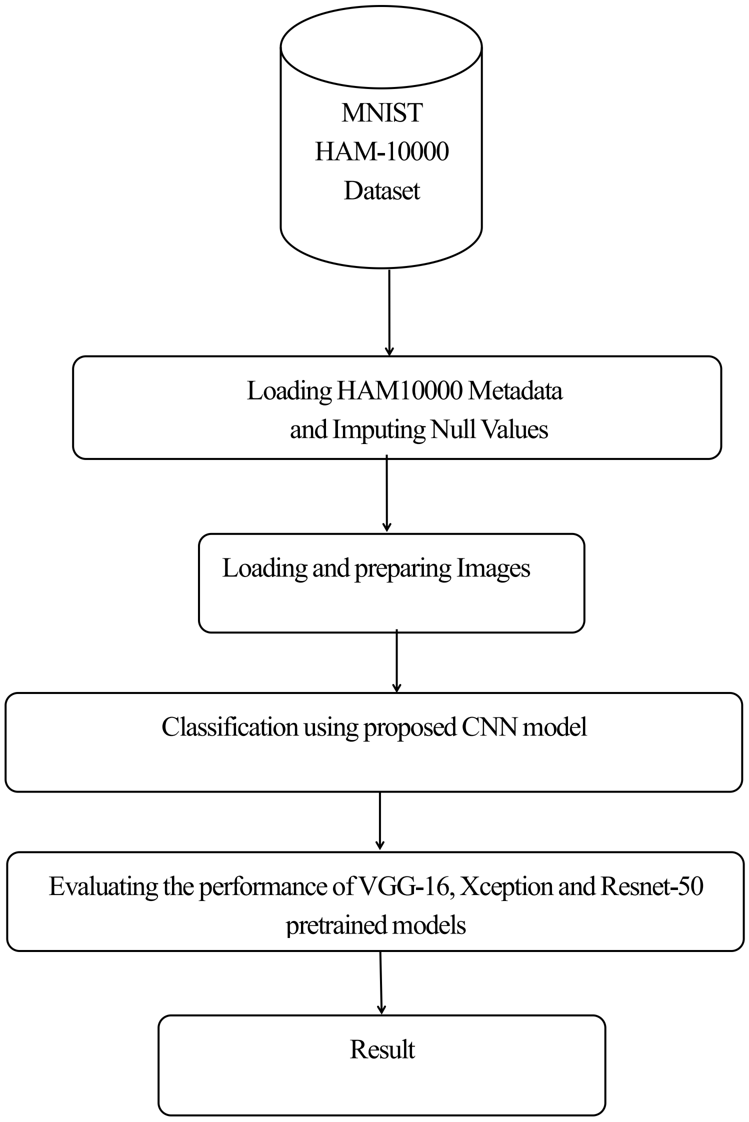

MNIST
HAM-10000
Dataset
Loading HAM10000 Metadata and Imputing Null Values
Loading and preparing Images
Classification using proposed CNN model
Evaluating the performance of VGG-16, Xception and Resnet-50 pretrained models
Result

Figure 12: Study workflow

3.1 Loading Metadata

Initially the metadata which is proved as a .csv file was loaded as pandas dataframe. The null values in the data were imputed. Then the given ‘dx’ column which provides the type of cancer was mapped to integer values. This column will be the target column of our neural network. Then complete image paths were loaded using the Image id column. Using these paths images were loaded.

3.2 Loading and Preprocessing Images

Images were loading to the dataframe in a new 'image_pixels' column corresponding to the image paths. The original dimension of images are 450x600x3 which TensorFlow can't handle, that's why images were resized to 224x224x3 as pretrained models such as ResNet, VGG-16 etc uses this format of images.

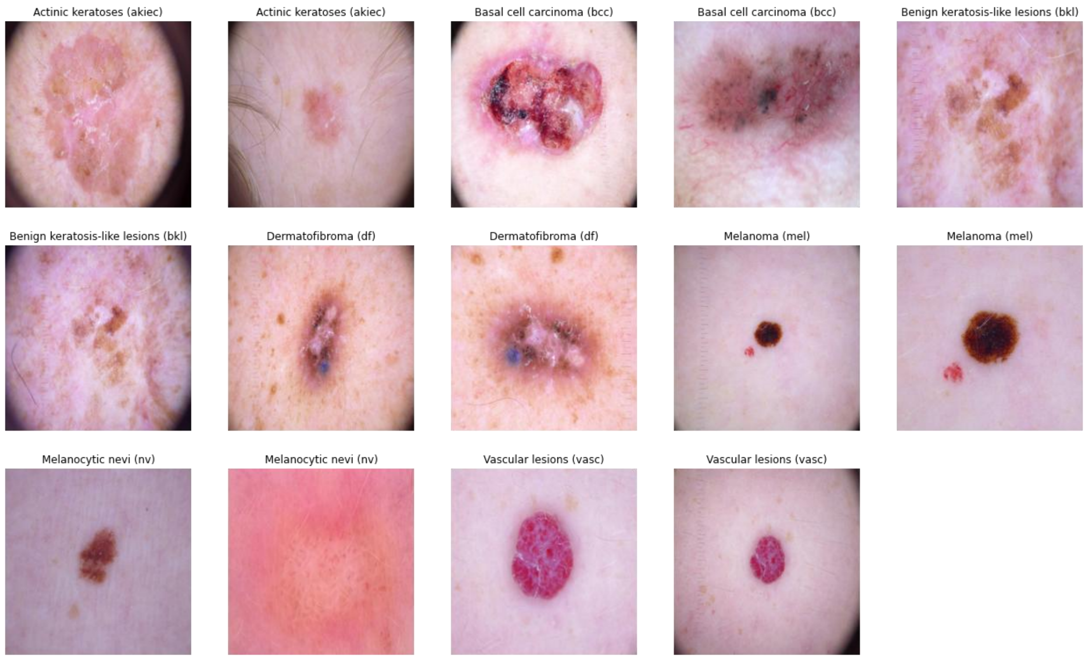

Actinic keratoses (akiec)
Actinic keratoses (akiec)
Basal cell carcinoma (bcc)
Basal cell carcinoma (bcc)
Benign keratosis-like lesions (bkl)
Benign keratosis-like lesions (bkl)
Dermatofibroma (df)
Dermatofibroma (df)
Melanoma (mel)
Melanoma (mel)
Melanocytic nevi (nv)
Melanocytic nevi (nv)
Vascular lesions (vasc)
Vascular lesions (vasc)

Figure 13: Samples of loaded images

3.3 Model Architecture

The convolutional neural network model proposed in this study includes: Convolution, Rectified Linear Unit (ReLU), Pooling, Flattening, fully connected layers, and Softmax function (Diagram below). At the initial step, several convolution filters are applied to the input image to determine features from the images. To develop the training speed, negative values are mapped to zero, and positive values remain unchanged in the ReLU step. The pooling step aims to simplify the output by performing nonlinear downsampling and decreasing the number of parameters that the network needs to train. In the flattening step, all two-dimensional arrays are modified into one single linear vector. This process is needed for fully connected layers to be used after convolutional layers. Fully connected layers can combine the entire local features of the previous convolutional layers. The procedure is completed with the application of the softmax activation function to provide the final classification output.

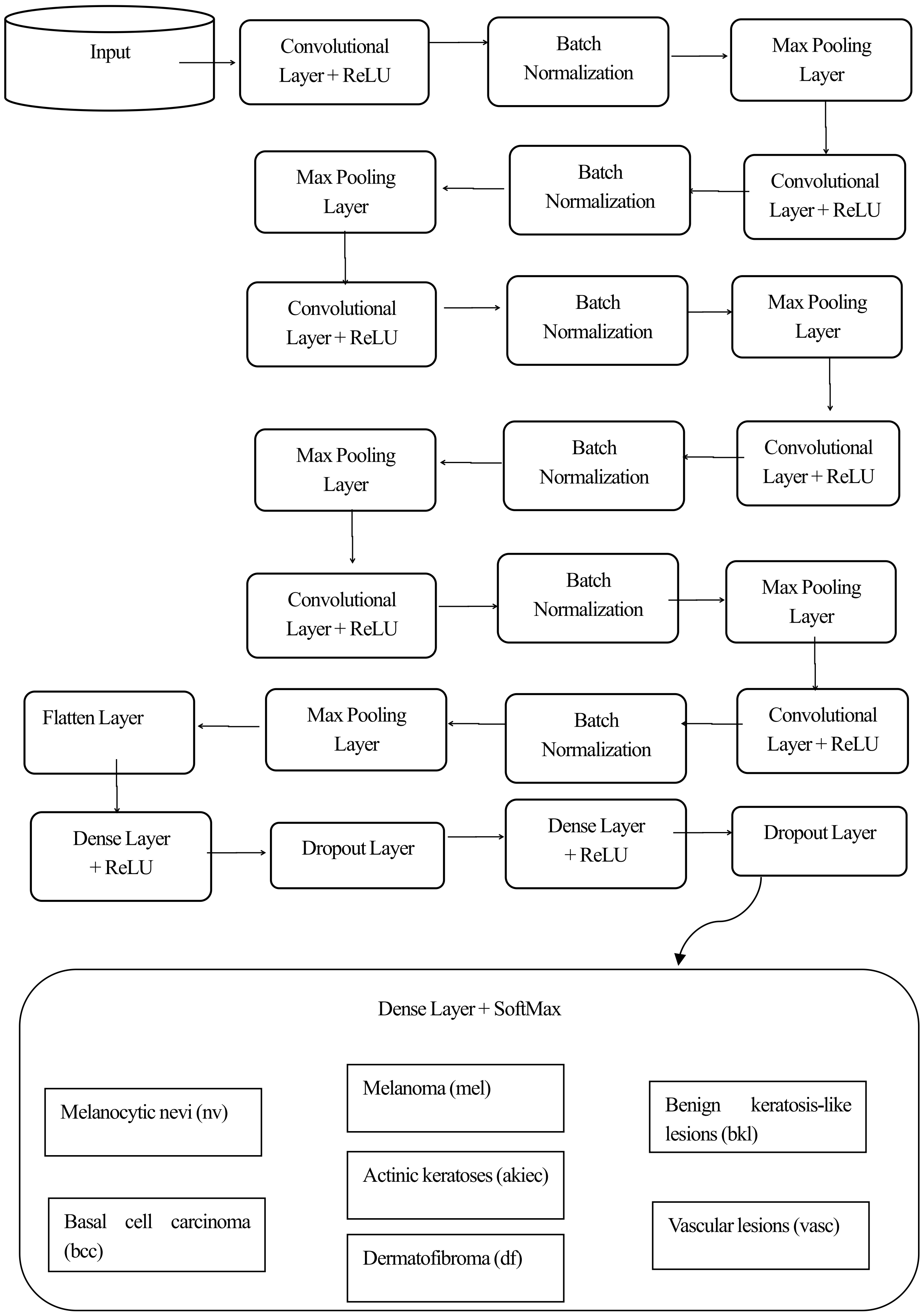
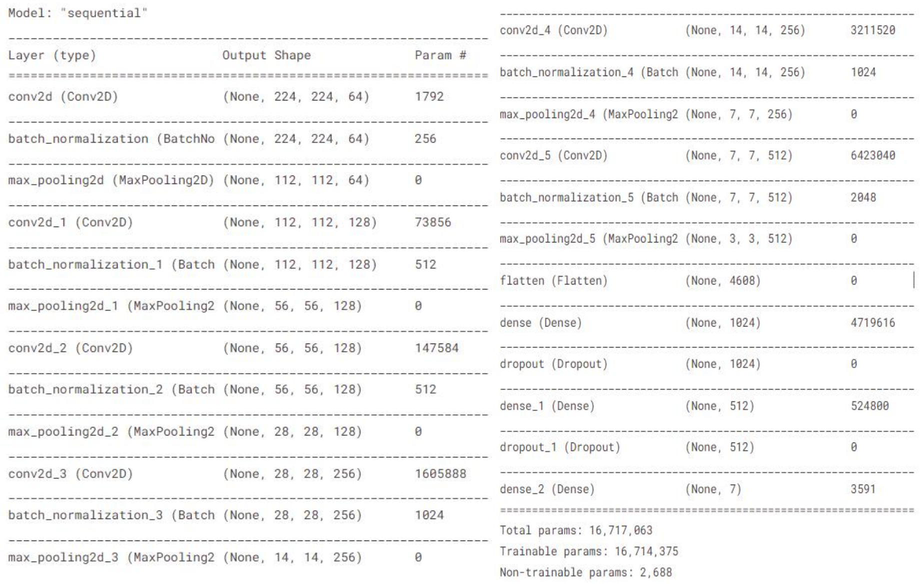


Figure 14: Model Architecture

The CNN model takes 224x224x3 images as input and passes them through six sets of Convolution2D, Batch Normalization and MaxPooling2D layers. In this process the images get converted into different dimensions and after the sixth MaxPooling layer the dimension of the image becomes 3x3x512. A Flatten layer is then used to convert the three-dimensional image to one dimensional vector. Then this vector goes through two Dense-Dropout layer blocks and finally gets classified at the last Dense layer. The total number of parameters generated in this process is 16,717,063 where 16,714,375 parameters are trainable and 2,688 parameters are non-trainable.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 224, 224, 64)      1792
_________________________________________________________________
batch_normalization (BatchNo (None, 224, 224, 64)      256
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 64)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 112, 112, 128)     73856
_________________________________________________________________
batch_normalization_1 (Batch (None, 112, 112, 128)     512
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 56, 56, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 56, 56, 128)       147584
_________________________________________________________________
batch_normalization_2 (Batch (None, 56, 56, 128)       512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 28, 28, 128)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 256)       1605888
_________________________________________________________________
batch_normalization_3 (Batch (None, 28, 28, 256)       1024
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 256)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 256)       3211520
_________________________________________________________________
batch_normalization_4 (Batch (None, 14, 14, 256)       1024
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 256)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 7, 7, 512)         6423040
_________________________________________________________________
batch_normalization_5 (Batch (None, 7, 7, 512)         2048
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 3, 3, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              4719616
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 3591
=================================================================
Total params: 16,717,063
Trainable params: 16,714,375
Non-trainable params: 2,688
```

Figure 15: Model built using Keras's Sequential class

3.4 Training the model

The model was trained in 30 epochs. Unequivocal Cross-entropy was utilized to estimate the loss. Stochastic Gradient Descent (SGD) with a knowledge rate of 0.001 was used to optimize the loss. A batch size of 32, steps per epoch of 5712//32 ≈ 178 and validation steps of 1311//32 ≈ 41 was used. The model took nearly 1 hour to be trained with a Nvidia K80 16GB GPU. The number of epochs of 30 has been standardized for this model because the learning rate degrades if the model is fed with too many iterations. Callback functions i.e., Early Stopping and Reducing Learning Rate on Plateau were also used to reduce overfitting. Model checkpoints was also used to save the best model while training. A gradual dropping of the validation accuracy of the model during the 17th and 24th epochs has been noticed, which is not at all a good sign. Furthermore, the model loses its effectiveness because of the over-fitting of the data. Due to this, the learning rate of the model decreases, which results in poor test accuracy.

3.5 Evaluating the model

The best performing model reached an accuracy of 99% on training set and 81% on test set. The categorical loss for training set was 0.0435 and for test set it was 0.6314. This model's performance was compared with the performances of conventional pretrained models, which revealed that the performance of this model is quite satisfactory.

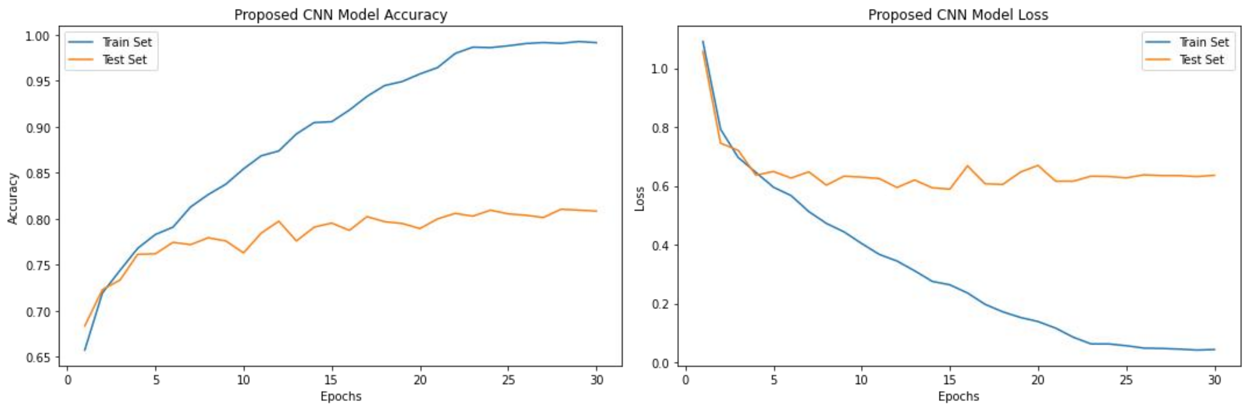


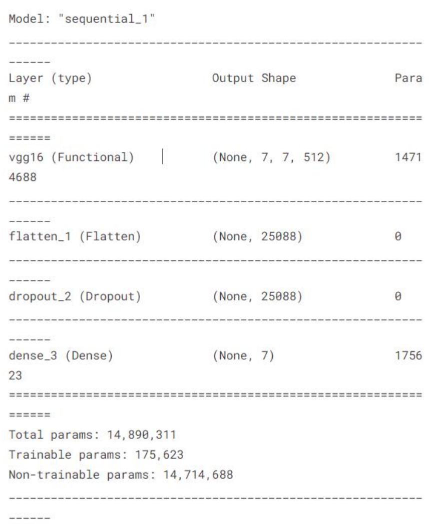
Figure 16: performance of CNN Model

The curves in Figure 16 clearly indicate that the model is overfitting. While training the model with accuracy monitoring, very unstable performance was observed. Thus, early stopping and the learning rate reduction functions were set to monitor the loss function. The ReduceLROnPlateau function reduced the learning rate to 0.0002 on 21st epoch and to 0.000004 on 27th epoch. However, these callback functions weren't capable of reducing the overfitting significantly but the helped the model achieve more stability. The test accuracy of 81% across seven different classes beats dermatologists with 5-6 years of experience.

3.6 VGG-16 training and evaluation

The VGG-16 model was implemented using Keras's API. Pretrained VGG-16 model weights were loaded and the VGG16 base model was extended by flatten, dropout and the final fully connected layer. RMSprop optimizer with an initial understanding rate of 0.00001 was used for training this model. This model was also trained for 30 epochs. The total number of parameters for this model was 14,890,311 of which 175,623 parameters were trainable and the remaining were pretrained. While training this model the learning rate reduced three times by a factor of 0.2.

The highest training and testing accuracies of this model were respectively 89% and 76%. Training this model for a longer period of time slightly increases the test accuracy but the loss curve starts to go up exhibiting unstable performances. Adam and SGD optimizers were also experimented with but they were unable to produce better performance.



```
Model: "sequential_1"
___________________________________________________________
______
Layer (type)                 Output Shape              Para
m #
===========================================================
======
vgg16 (Functional)     |     (None, 7, 7, 512)         1471
4688
___________________________________________________________
______
flatten_1 (Flatten)          (None, 25088)             0
___________________________________________________________
______
dropout_2 (Dropout)          (None, 25088)             0
___________________________________________________________
______
dense_3 (Dense)              (None, 7)                 1756
23
===========================================================
======
Total params: 14,890,311
Trainable params: 175,623
Non-trainable params: 14,714,688
___________________________________________________________
______
```

Figure17: structure of the VGG-16 model

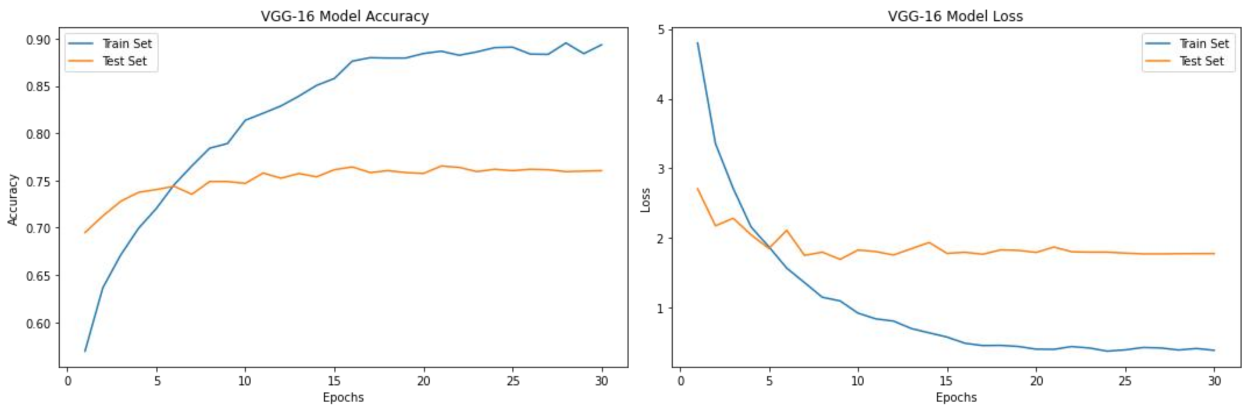
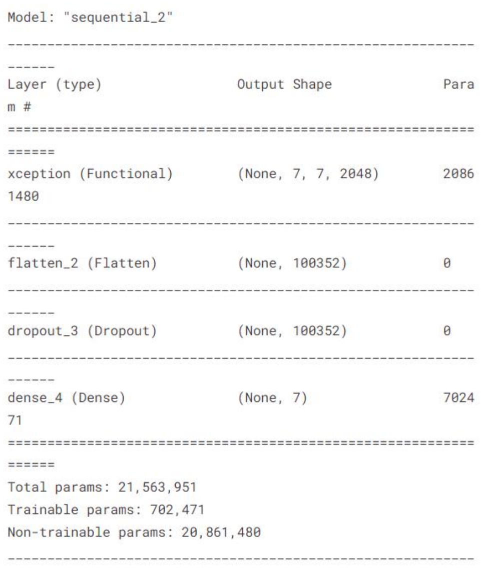


Figure18: performance of the VGG-16 model

3.7 Xception training and evaluation

The Xception model was also implemented using Keras and pretrained model weights. Modifications similar to VGG-16 were made to this model also. Categorical cross-entropy loss and RMSprop optimizer with an initial learning rate of 0.00004 was used. The number of trainable parameters for this model was 702,471 whereas the total number of parameters was 21,563,951.

This model's performance was the most unstable compared to the other models trained during this study. The loss for both training and testing sets were very high (>= 9.1564). The zigzagging curve of accuracy shows the performance of this model was unsatisfactory. The highest training and testing accuracies achieved by this model was 72% and 73% respectively. SGD and Adam optimizers were also used with an aim to decrease the unstable behavior of this model but none could solve this problem.



```
Model: "sequential_2"
_________________________________________________________________
______
Layer (type)                 Output Shape              Para
m #
=================================================================
======
xception (Functional)        (None, 7, 7, 2048)        2086
1480
_________________________________________________________________
______
flatten_2 (Flatten)          (None, 100352)            0
_________________________________________________________________
______
dropout_3 (Dropout)          (None, 100352)            0
_________________________________________________________________
______
dense_4 (Dense)              (None, 7)                 7024
71
=================================================================
======
Total params: 21,563,951
Trainable params: 702,471
Non-trainable params: 20,861,480
_________________________________________________________________
```

Figure19: Xception model structure

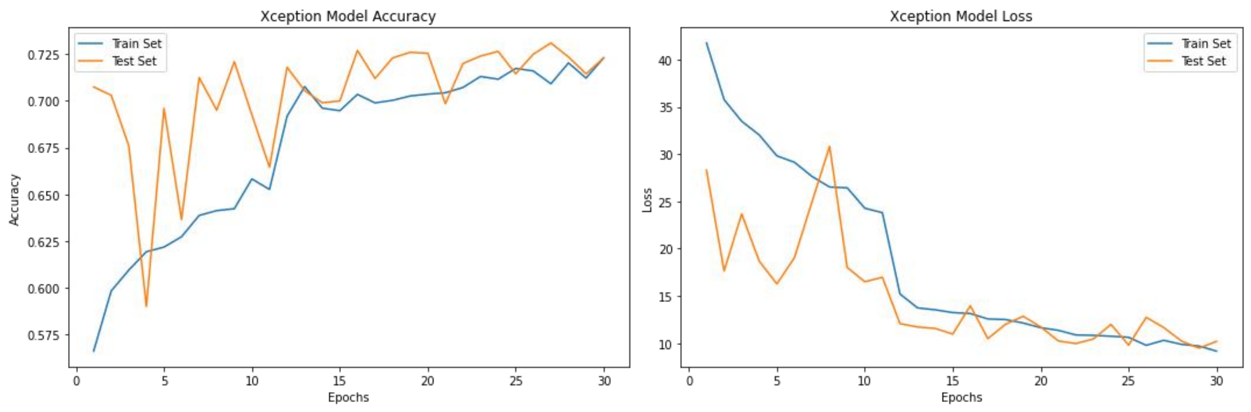
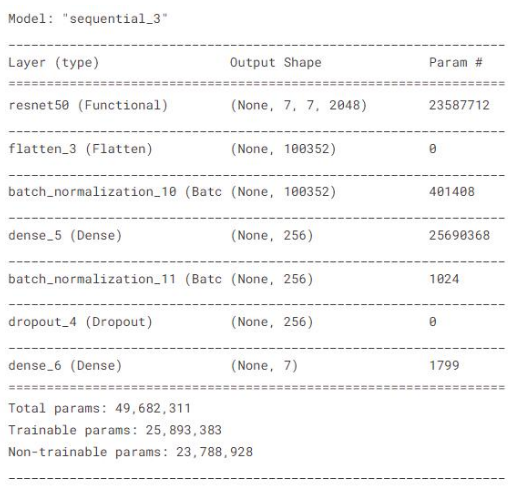


Figure20: Xception model performance

3.8 ResNet-50 training and evaluation

The ResNet-50 model was fine tuned with two normalization and fully connected layer blocks. The hidden fully connected layer had 256 neurons which after passing through a dropout layer arrives at the final classification layer. The total number of parameters for this model was 49,682,311 of which 25,893,383 were trainable. The model was trained for 30 epochs. Adam optimizer with an initial understanding rate of 0.001. During training the learning rate reduced four times on 8th, 14th , 20th and 26th epochs respectively and the final learning rate was 1.6e-06.

The highest training and testing accuracies for this model were 99% and 79% respectively. The curves in Figure23 clearly show that the model is overfitting. The loss on testing set starts to increase after 9th epoch which is not a good sign. This causes the learning rate to drop several times and finally the loss stabilizes around 1.20. Though this model seemed to perform well in terms of accuracy, it has failed decrease the loss adequately.



```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 7, 7, 2048)        23587712
_________________________________________________________________
flatten_3 (Flatten)          (None, 100352)            0
_________________________________________________________________
batch_normalization_10 (Batc (None, 100352)            401408
_________________________________________________________________
dense_5 (Dense)              (None, 256)               25690368
_________________________________________________________________
batch_normalization_11 (Batc (None, 256)               1024
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_6 (Dense)              (None, 7)                 1799
=================================================================
Total params: 49,682,311
Trainable params: 25,893,383
Non-trainable params: 23,788,928
_________________________________________________________________
```

Figure21: ResNet-50 structure

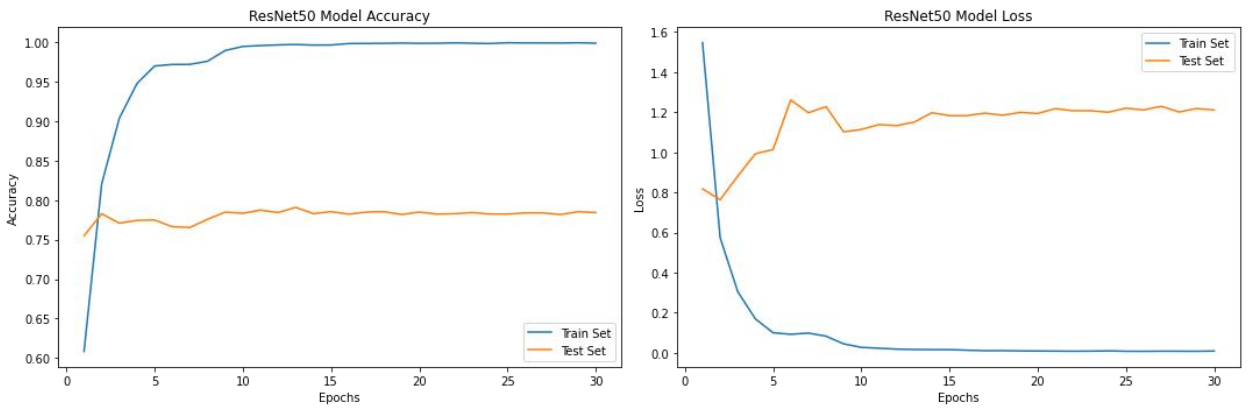


Figure22: ResNet-50 performance

3.9 Performance Comparison

The performance of four deep learning models for skin cancer classification was evaluated based on input data size, model complexity, accuracy, model loss, and accuracy curves. The number of convolutional layers and the parameters of each model are shown in Table 1. From the table, the proposed CNN model possesses a minimal number of convolutional layers and network parameters compared with the other pre-trained models. The Xception model is the deepest model and has 53 times more network parameters than the proposed model. The complexity of the proposed CNN model is lower than other pre-trained models in terms of its number of parameters. ResNet50 model has a three-times more significant number of parameters than the proposed CNN model. Moreover, the depth and number of network parameters of the Xpection and ResNet-50 models are higher than that of proposed model. However the proposed model has more parameters than the VGG-16 model.

Table 1: Deep learning models with the number of convolutional layers and parameters

Deep Learning Model	Number of convolutional layers	Number of parameters
Proposed CNN Model	6	16,717,063
VGG-16	13	14,890,311
Xception	71	21,563,951
ResNet-50	48	49,682,311

Table 2: Models training accuracy and validation accuracy

Deep Learning Model	Training Accuracy	Testing Accuracy
Proposed CNN Model	99%	81%
VGG-16	89%	76%
Xception	72%	73%
ResNet-50	99%	79%

In the medical field, we always need something that can predict or detect things accurately, and less accuracy can play with a patient's life. The dataset on different pre-trained models has trained and built a CNN model from scratch. If we demonstrate Table 2, we can see that the proposed CNN model gives much higher accuracy both in training and testing than the other pre-trained models. However, other pre-trained models provide a good result but are not good enough to beat the proposed CNN model as it is a matter of detecting the skin cancer, which is the most dangerous illness in the modern world. That is why the highest accuracy must be considered.

We also know that a loss function in Machine Learning measures how accurately a model can predict the expected outcome, i.e., the ground truth. After observing Table 3, we can see that the proposed CNN model gives less loss than other pre-trained models. It is a reason that the proposed CNN model can predict more accurately than other pre-trained models.

Table 3: Models result after evaluation

Deep Learning Model	Loss	Accuracy
Proposed CNN Model	0.6314	81%
VGG-16	1.7814	76%
Xception	11.6497	73%
ResNet-50	1.1508	79%

Observing Figure 22, we can clearly see that the proposed CNN model outperforms the other pretrained models. The curves were generated on same scales for Y axis. From the loss curves it can be derived that the loss on proposed CNN model is better than other models. Though ResNet-50 gives a similar performance to the proposed model, after comparing both loss and accuracy curves we can clearly see that the proposed CNN model performs better than pretrained ResNet-50 model. VGG-16 having less parameters gives a stable and somewhat satisfying result. Whereas, Xception being the deepest model performed worst. This may have been caused by the highly specifically pretrained parameters of Xception model which gives very high loss compared to the other models.

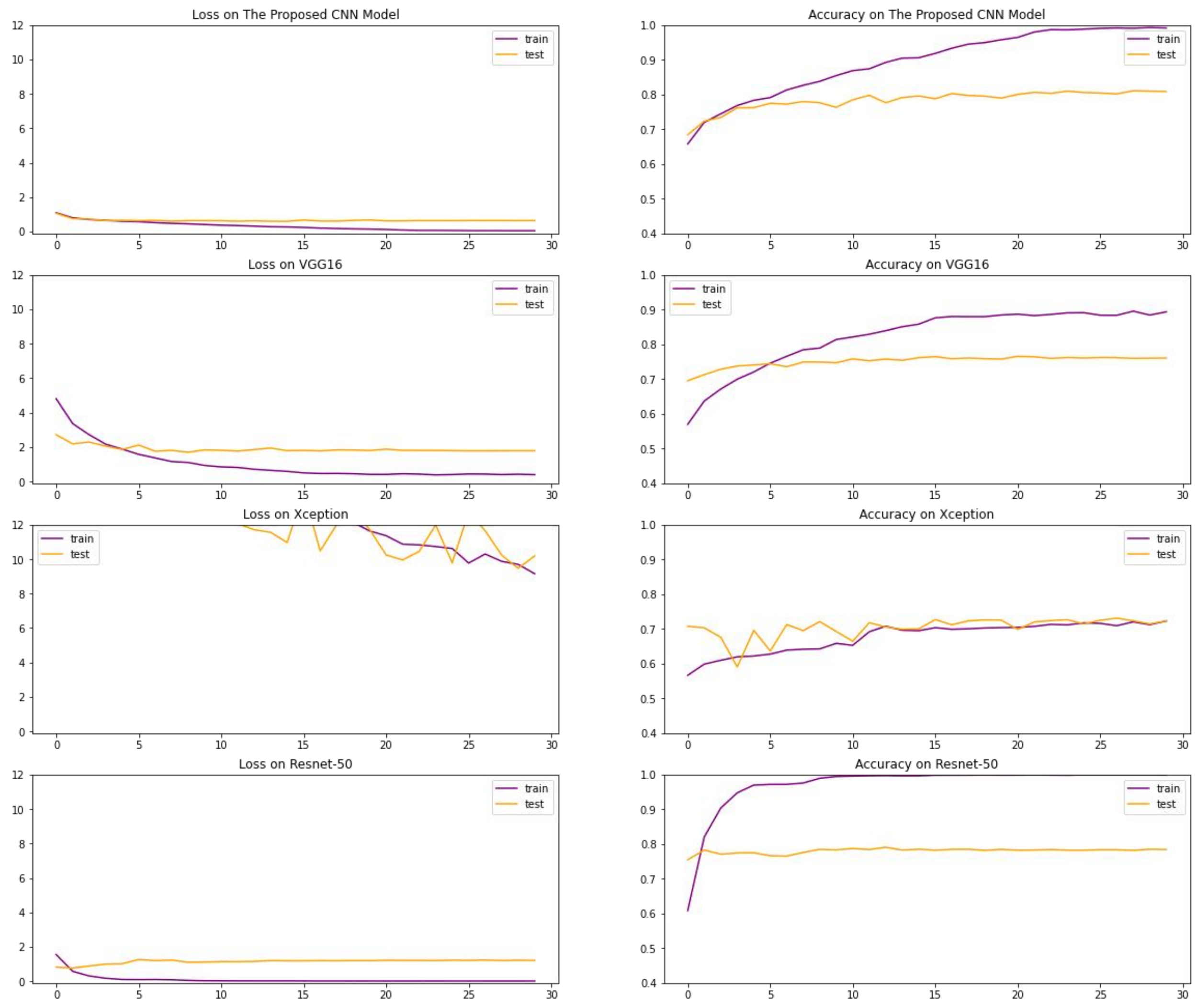


Figure 23: Comparing loss and accuracy curves

4. Web Application

4.1 Development

A web application was developed using the CNN model built during this study to assist dermatologists in classification of skin lesions. Django framework of Python programming language was used to build the backend and route the website. HTML, CSS and Bootstrap 5 was used to design the frontend.

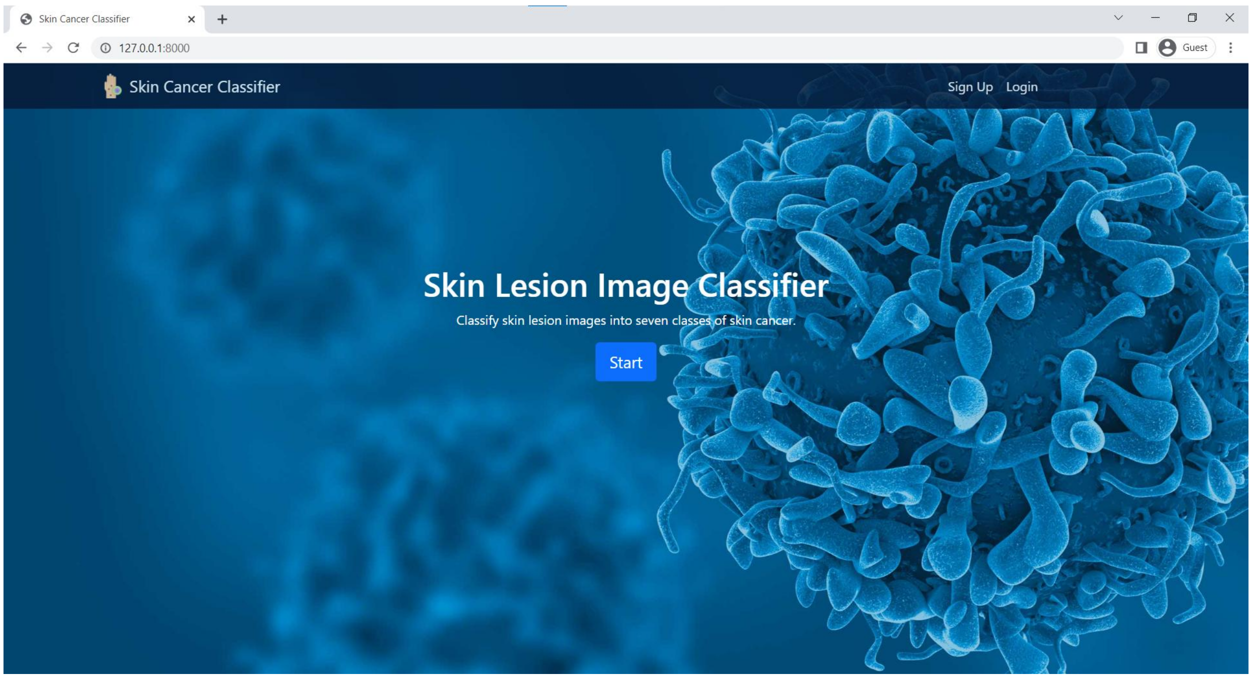

Skin Cancer Classifier
127.0.0.1:8000
Guest
Skin Cancer Classifier
Sign Up
Login
Skin Lesion Image Classifier
Classify skin lesion images into seven classes of skin cancer.
Start

Figure 24: Homepage of the web application

4.2 Backend

Django is a well-developed and popular framework of Python for web development. Initially, necessary tables were created in the database. Django's default SQLite database was used for this purpose. Every classification performed in the website is stored in the database along with the image that was classified for future development of the classifier model's performance. The django.db.model.Models class was extended to create the database tables. Necessary view functions were implemented to render the web pages. Only logged in users were permitted to use the classifier. Django's built-in user and authentications were used for these purposes. The CNN model was loaded in the website using the Keras and Tensorflow. Classification results of all skin lesions are also stored in the database. These are stored to produce a better dataset and train a better CNN model for more accurate predictions.

4.3 Frontend

Frontend was developed using HTML, CSS and Bootstrap 5. These assisted in creating a beautiful and user-friendly frontend. As it a simple prototype, only six webpages were required for this web application. The user interfaces were made as simple as possible so that doctors can use it easily without any unnecessary difficulties. Once the user is logged in a form is presented to the user where the image can be uploaded which is to be classified. By pressing the check button the Django view function classification of the image using the trained CNN model is triggered. Then the user is taken to the results page where the predictions are presented in a bar chart. The bar chart indicates the probability of the image belonging to seven classes of skin cancer.

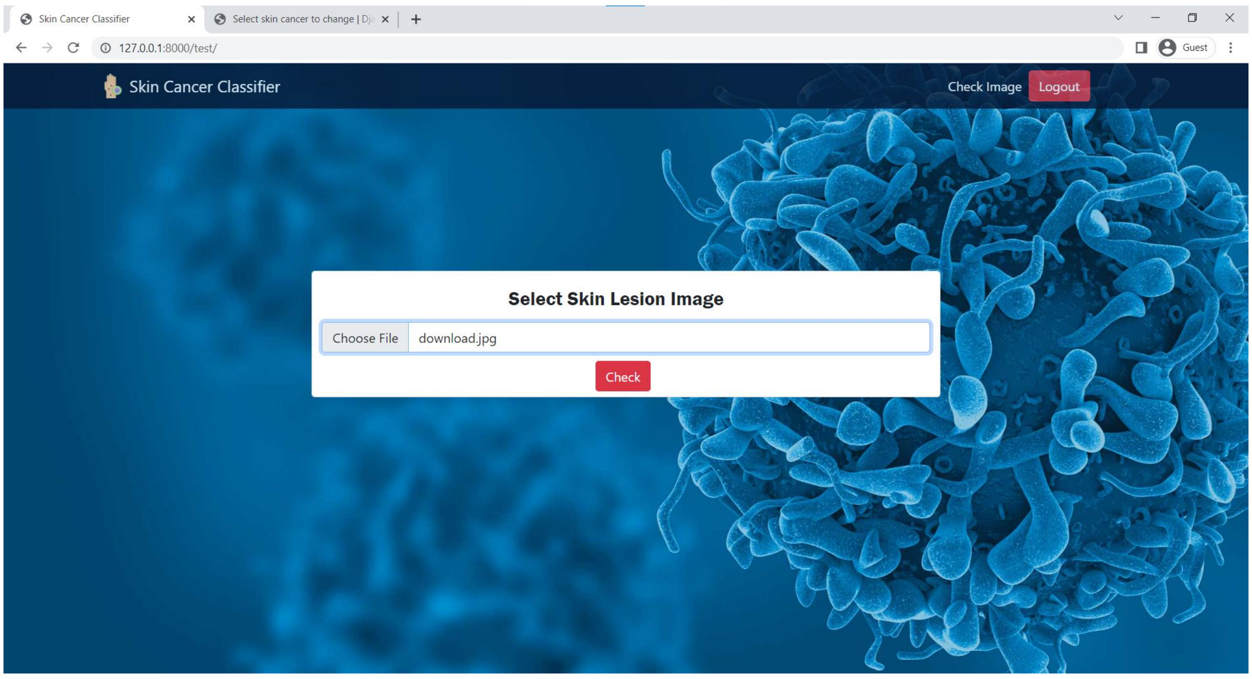


Figure 25: selecting skin lesion image

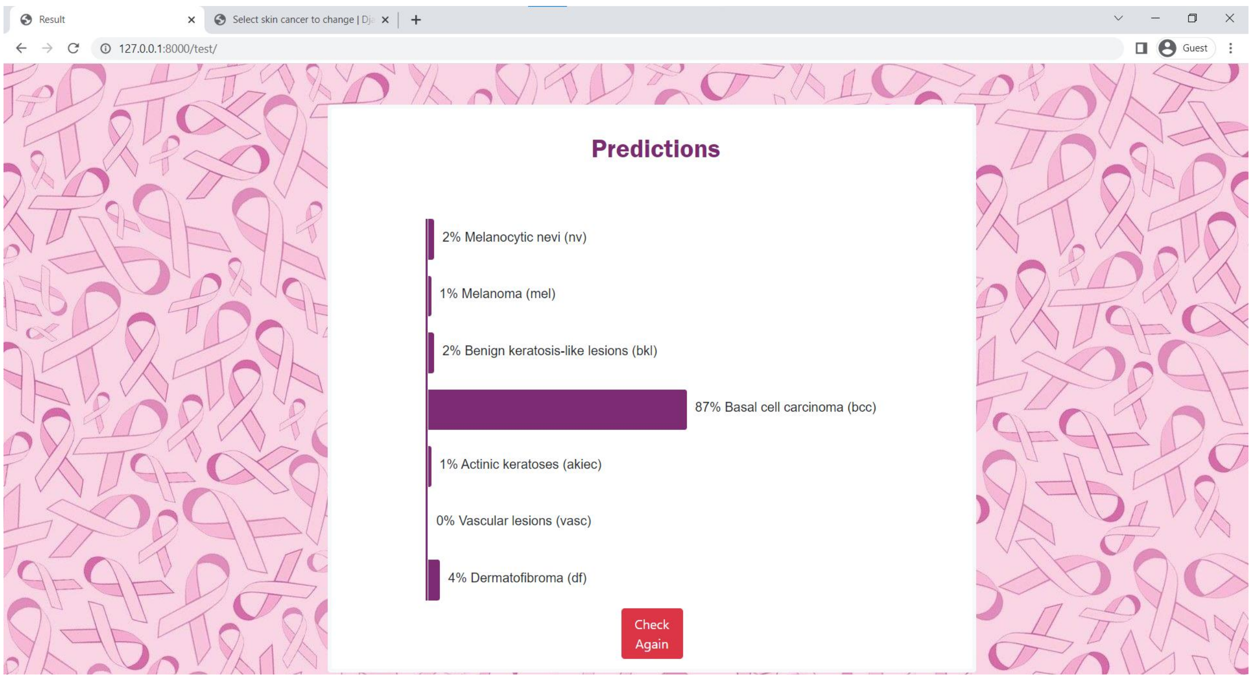

Result
Select skin cancer to change | Dj
127.0.0.1:8000/test/
Guest
Predictions
2% Melanocytic nevi (nv)
1% Melanoma (mel)
2% Benign keratosis-like lesions (bkl)
87% Basal cell carcinoma (bcc)
1% Actinic keratoses (akiec)
0% Vascular lesions (vasc)
4% Dermatofibroma (df)
Check Again

Figure 26: Prediction results

5. Conclusion and Future Work

5.1 Conclusion

In conclusion, this paper represents a new approach to classifying skin cancer lesions from images. First, we find the region of interest in dermoscopy images and cropped them using the image edge detection technique. Second, it provided an efficient methodology by proposing a CNN model. For sophisticated and accurate results, the neural network requires a large amount of data to train. However, the experimental result shows that even with low computational power, this model can attain satisfactory accuracy, and the accuracy rate is excellent compared to VGG-16, Xception and ResNet-50 models. The proposed CNN model training time is faster than the VGG-16, Xception and ResNet50 models. Subsequently, the proposed CNN model needs fewer computational specifications as it takes less execution time.

Moreover, the proposed CNN model accuracy is better than VGG-16, Xception and ResNet-50 models. The proposed system can play a prognostic significance in the classification of skin cancer for both patients and physicians. To further boost the model efficiency, comprehensive hyper-parameter tuning and a better pre-processing technique can be conceived. The proposed system is for categorical classification problem, which can classify the cancer type into seven classes. The plotting of the model accuracy and loss graph were made to compare the proposed Model with VGG-16, Xception and ResNet50 models. Comparatively, the proposed CNN model can detect and classify skin cancer better than other proposed pre-trained models. Also, this proposed system can play an influential role in the early diagnosis of dangerous diseases in other clinical domains related to medical imaging, particularly lung cancer and breast cancer, whose mortality rate is very high globally. We can prolong this approach in other scientific areas where there is a problem in the availability of extensive data, or it can be used with different transfer learning methods with the same proposed technique.

5.2 Future Work

In future work, I will combine multiple models to build a meta-model for better performance. This process is known as Ensemble Learning. The prediction levels are much more accurate with the use of the Ensemble Learning technique. However, the implementation of this meta-model requires a high-performance Graphics Processing Unit (GPU) in addition to a CPU that demands high computational power, cost, and time. The future work lies in the implementation of Skin Cancer Classification using Ensemble learning.

References

[1] "Early diagnosis of malignant melanoma," [Online]. Available: https://www.clinicaladvisor.com/features/early-diagnosis-of-malignant melanoma/article/305429/.

[2] M. Celebi, Y. Aslandogan, W. Stoecker, H. Iyatomi, H. Oka and X. Chen, "Unsupervised border detection in dermoscopy images," *Skin Research and Technology,* vol. 13, no. 4, pp. 454-462, 2007.

[3] M. Fleming, C. Steger and e. al., "Techniques for a structural analysis of dermatoscopic imagery," *Computerized Medical Imaging and Graphics,* vol. 22, no. 5, pp. 375-389, 1998.

[4] G. Argenziano, H. Soyer and e. al., "Dermoscopy of pigmented skin lesions: results of an agreement meeting through the internet," *Journal of the American Academy of Dermatology,* vol. 48, no. 5, pp. 679-693, 2003.

[5] R. Kenet and T. Fitzpatrick, " Reducing mortality and morbidity of cutaneous melanoma: a six year plan. b). identifying high and low risk pigmented lesions using epiluminescence microscopy," *The Journal of Dermatology,* vol. 21, no. 11, pp. 881- 884, 1994.

[6] S. Menzies, C. Ingvar, K. Crotty and e. al., "Frequency and morphologic characteristics of invasive melanoma lacking specific surface microscopy features," *Archives of Dermatology,* vol. 132, pp. 1178-1182, 1996.

[7] H. Pehamberger, A. Steiner and K. Wolff, " In vivo epiluminescence microscopy of pigmented skin lesions. i. pattern research of pigmented skin lesions," *Journal of the American Academy of Dermatology,* vol. 17, no. 4, pp. 571-583, 1987.

[8] G. Argenziano, G. Fabbrocini, V. De Giorgi, P. Carli, E. Sammarco and M. Delfino, "Epiluminescence microscopy for diagnosing doubtful melanocytic skin lesions: Comparison of the ABCD fule of dermatoscopy and a new 7-point checklist based on pattern analysis," *Archives of Dermatology,* vol. 134, no. 12, pp. 1563-1570, 1998.

[9] W. Sterry and R. Paus. Thieme clinical companions dermatology. In Thieme Verlag. Stuttgart, New York., 2006.

[10] Skin Cancer Foundation. Skin cancer facts and statistics. [online]. http://www. skincancer.org/skin-cancer-information/skin-cancer-facts, 2016. [Accessed: January 12 2017].

[11] American Cancer Society. Skin cancer prevention and early detection. [online]. http:// www.cancer.org/acs/groups/cid/documents/webcontent/003184-pdf.pdf, 2015.

[12] A. G. Goodson and D. Grossman. Strategies for early melanoma detection: approaches to the patient with nevi. [online]. http://www.cancer.org/acs/groups/cid/documents/ webcontent/003184-pdf.pdf, 2009.

[13] R.L. Siegel, K.D. Miller, and A.Jemal. Cancer statistics. In A cancer journal for clinicians, 66(1), pp.7-30., 2016.

[14] F. Nachbar, W. Stolz, T. Merkle, A.B. Cognetta, T. Vogt, M. Landthalerv, P. Bilek, O. Braun-Falco, and G. Plewig. The abcd rule of dermatoscopy: high prospective value in the diagnosis of doubtful melanocytic skin lesions. Dermatology Journal of the American Academy, 30(4), pp.551-559., 1994.

[15] "Menzies Method," [Online]. Available: https://dermoscopedia.org/Menzies_Method.

[16] G. Argenziano. Seven-point checklist of dermoscopy revisited. In British Journal of Dermatology 164, no. 4 (2011): 785-790., 2011.

[17] S.W. Menzies. A process for the diagnosis of primary cutaneous melanoma using surface microscopy. In Dermatologic clinics, 19(2), pp.299-305., 2001

[18] V. Nair, G. Hinton, Rectified linear units improve restricted boltzmann machines, ICML, 2010.

[19] H. Robbins, S. Munro, A Stochastic Approximation Method, Ann. Math. Stat., 22 (1951), 400-407.

[20] Categorical crossentropy loss function Peltarion Platform.https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy

[21] G. Hinton, Neural networks for machine learning online course lecture 6a, Coursera.Available from: http://www.cs.toronto.edu/tijmen/csc321/slides/lecture_slides_lec6.pdf.

[22] Luo et al. (2019). Adaptive Gradient Methods with Dynamic Bound of Learning Rate. In Proc. of ICLR 2019.

[23] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv: 1409.1556.

[24] F. Chollet, “Xception: Deep learning with depthwise dividable convolutions,” in 2017 IEEE Conf. on Computer Vision and Pattern Recognition, Honolulu, HI, pp. 1800–1807, 2017.

[25] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, Proceedings of the IEEE forum on computer vision and pattern recognition, 2016.

Statement

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得四川大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

I hereby declare that the academic thesis that I submit is the research work and results that I conducted and achieved under the guidance of my adviser. To my knowledge, except where especially noted and acknowledged, this thesis doesn’t contain any research findings that have been published or written by others, or any materials that have been used to obtain degrees or certificates from Sichuan University or other Educational Institutions. Any contribution made by the comrades working with me to this research has been clearly illustrated and acknowledged in this thesis.

本学位论文成果是本人在四川大学读书期间在导师指导下取得的，论文成果归四川大学所有，特此声明。

I also hereby declare that the results of this academic thesis are achieved under the guidance of my adviser, during my study in Sichuan University and they are owned by Sichuan University.

学位论文作者（签名）Saykat ______________________

Thesis Author (Signature) Saykat ______________________

论文指导教师（签名）

Thesis Advisor (Signature) ______________________

年　月　日

(mm)　(dd),　　(yyyy)

Acknowledgement

I want to reveal my utmost gratitude and respect for my advisor, associate professor Chuan Li, for his support and guidance throughout my Bachelor's career. His dedication and expertise were unyielding sources of inspiration and encouragement. Without his guidance and continued help, this thesis would not have been a success for me. I would also like to present my most sincere respect and thanks to the program coordinator, Ms. Ma Yi, for her understanding and commitment to supporting me develop my passion and knowledge. I want to thank all the thesis committee members for offering guidance and feedback for all stages of this thesis work. I am grateful to the Sichuan University, the College of Computer Science, and the College of Software Engineering, without whom this thesis would not have been an achievement.

I also thank my family and friends for their endless love and support throughout my life.