The University of Nottingham

Faculty of Engineering

Department of Electrical and Electronic Engineering

< Personalized Skin Cancer Detection based on Cloud Computing - BEng>

AUTHOR	:	Gan Jia Lian
ID	:	20007597
SUPERVISOR	:	Dr. Hermawan Nugroho
MODERATOR	:	Dr. Ananda Shanmugam
DATE	:	22/09/2020

Third year project thesis submitted in partial fulfillment of the requirements of the degree of **Bachelor of Engineering**

List of Figure

List of Table

Nomenclature

ML	Machine Learning
CNN	Convolutional Neural Network
ANN	Artificial Neural Network
DNN	Deep Neural Network
akiec	Actinic Keratosis
bcc	Basal Cell Carcinoma
bkl	Benign Keratosis-Like Lesion
df	Dermatofibroma
mel	Melanoma
nv	Melanocytic Nevi
vasc	Vascular Lesion
IDE	Integrated Development Environment
API	Application Programming Interface
ReLU	Rectified Linear Unit
UV	Ultraviolet
M	Millions

Abstract

Today, the depletion of the ozone layer causes the UV radiation on the earth become more harmful and dangerous to humans and thus skin cancer become more common in human daily life. Four popular convolutional neural networks (CNNs) which are InceptionV3, Xception, ResNet50V2 and DenseNet121 from Keras applications library proposed in this paper for skin cancer classification. About 10000 images of seven categories of skin cancer datasets from Kaggle website are used for the models to train and evaluate for the performance of CNNs on images classification. The trained models then implemented in TensorflowJS framework for web application and analysed again for the classification performance in web application. 105 skin cancer images with 15 images in each category randomly choose from the dataset is used to perform testing on the trained models using Jupyter Notebook and web application. The classification results in Jupyter Notebook show the average accuracy of InceptionV3, Xception, ResNet50V2 and DenseNet121 are 0.87, 0.80, 0.85 and 0.94 respectively, while classification results in web application are 0.92, 0.91, 0.90 and 0.94 respectively. DenseNet121 has the best performance among the trained models with the highest accuracy and no error different in classification performance in Jupyter Notebook and web application.

Table of Contents

Chapter 1 – Introduction

1.1 Background

Today, skin cancer is a killing disease, and it was ignored by most of the people in daily life because most of them don't pay much attention to their skin condition and there are many areas that are not easy to examine, such as the back of the body, the bottom of the foot or the top of the head.[1] Moreover, some religion such as Muslim has their religious beliefs that they cannot simply show their skin to the others and this will cause more people only willing to seek for doctors when the skin cancer becomes severe and easily lead to death. Besides that, the depletion of ozone layer increases the exposure of harmful UV radiation to the earth, causing a total of 40% of the total cancer cases. Skin cancer usually at early stage will not spread to other organs or cells which make it hard to detect.[2]

With the rapid development of technology, lot of various type of data such as images, videos and texts are flooding the internet, and this had stimulated the development of machine learning and deep learning in different applications such as image classification, object recognition and text recognition. Deep learning such as convolutional neural network is adopted for skin cancer classification which can achieve the dermatologist accuracy level or even better for classification of skin cancer.[2] Four different CNN transfer learning models are proposed in this research and aim for classifying seven categories of skin cancer including Bowen's disease (akiec), basal cell carcinoma (bcc), benign keratosis-like lesions (bkl), dermatofibroma (df), melanoma (mel), melanocytic nevi (nv) and vascular lesions (vasc).[3] The transfer models are fine-tuned for the fully connected layer for a suitable output. Next, the input image data from Kaggle's dataset will pre-processed and fed into the CNN models for feature extraction and learning.

1.2 Aims and Objectives

The research is aimed to create a cloud computing framework like web application or mobile application which are compatible with the transfer learning model of Tensorflow integrated Keras library in Python programming language.

The objective of this paper:

- Train a skin cancer detection algorithm of the transfer learning with an accuracy of 90% above and minimal loss of 40% below using Keras applications.
- Implementation of trained model weight into web application with TensofloWJS framework which is functionable and user-friendly for people to upload their skin image and doing prediction or classification of skin diseases.

1.3 Deliverables

The proposed outcomes of the project are stated as follows:

1. A complete cloud-based web application for skin cancer classification on image uploaded by users.

1.4 Industrial Relevance

In real life applications, implementation of deep learning algorithm is a simple and straight forward tool for image classification, object detection and many more. With uses of deep learning, the algorithms show the physician-level diagnostic performance by learning and training on thousands of lesions images until the algorithm recognize the disease states. Besides that, these data images will be collected by expert and updated continuously for the algorithm can improve more further. Now, trained vision algorithms can be implemented on medical device for collecting data such as images or videos. In future, medical device may replace by edge devices such as smartphone with deep learning algorithm for collecting data without dermatoscope attachments. However, the deep learning algorithm is still not ready to deploy in clinical use. Especially for melanoma classification cases, there are only a few research studies done compared to the performance of algorithms with physicians at the task of classification. Moreover, the data used for training deep learning algorithms can different with the data processed by the physicians in current clinical environment. Deep learning algorithm also suffer from easy saturation in training process, overfitting problems and even miscorrelations of datasets can affect the performance of the algorithm. As an example, the training images contain the marking which is used by dermatologists will misdirect the algorithms to learn the presence of the marking, resulting inaccurate in the classification on the lesion images without artifacts. [4]

To overcome the challenges, deep learning algorithms require validation by testing a functioning system against the clinical analysis of dermatologists. The varies in the system, such as use different edge devices and dermatocope attachments also need for validation. Next, the deep learning algorithm need to be certified for classification of all diseases within a particular class to prevent wrong prediction. If these challenges and problems can be solved, deep learning algorithms will be an important tool in general clinical use by becomes a second opinions for dermatologists to confirm the diseases in difficult cases. Besides that, patients can perform self-analysis at home without pay an unnecessarily clinical visit and allow them take cautious early on the skin diseases which may missed.[4]

Chapter 2 - Literature Review

2.1 Overview of Machine Learning, ML

Artificial intelligent system enables the computers to solve or perform complex tasks such as knowledge representation, reasoning, learning planning, perception and communication independently or with minimal human intervention. In the artificial intelligent system, all the artificial capabilities such as image recognition, image classification, natural language processing, etc are rely on the machine learning based algorithm. Machine learning plays an important in artificial intelligent because the algorithms or networks can learn and refer to the dataset that fed into it, then perform the required artificial capabilities. Moreover, machine learning can solve the limitations of humans. Machine learning is a computer program which can improve their performance and provide more reliable decision by continuous learning and training process from previous computations and new datasets. Thus, machine learning has good applicability in many areas, such as fraud detection, marketing analysis, image classification and many more.[5]

In machine learning, there are 3 different concepts or classes including machine learning algorithm, artificial neural networks and deep neural networks as shown in Figure. 1.

Depending on the learning task, machine learning can provide different classes of applications in machine learning algorithm, with different specification and purpose. As example, support vector machine, decision tree, ANNs, etc are machine learning algorithms.

Artificial neural networks have a flexible architecture which allows modification for a variety of contents compared with machine learning algorithm and deep neural networks. In artificial neural networks, there are zero or more hidden layers in the networks architecture for processing the learning of a non-linear mapping between input and output.

Deep neural networks have more complex and deeper network architecture which consist of more than one hidden layer. The core ability of DNN is known as deep learning, which allow the deep networks to be fed with raw input data and discover a representation automatically for a required learning task. Deep learning usually used for application which required process large and high-dimensional dataset such as image, text, audio, video, etc.

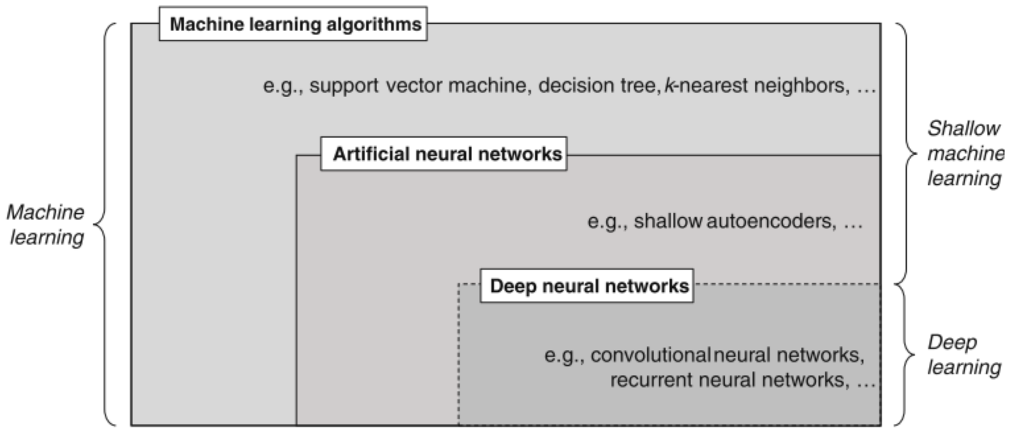

Machine learning algorithms
e.g., support vector machine, decision tree, k-nearest neighbors, …
Artificial neural networks
e.g., shallow autoencoders, …
Deep neural networks
e.g., convolutional neural networks, recurrent neural networks, …
Machine learning
Shallow machine learning
Deep learning

Figure 1 Machine Learning Concepts & Classes (reproduced from [5] *)*

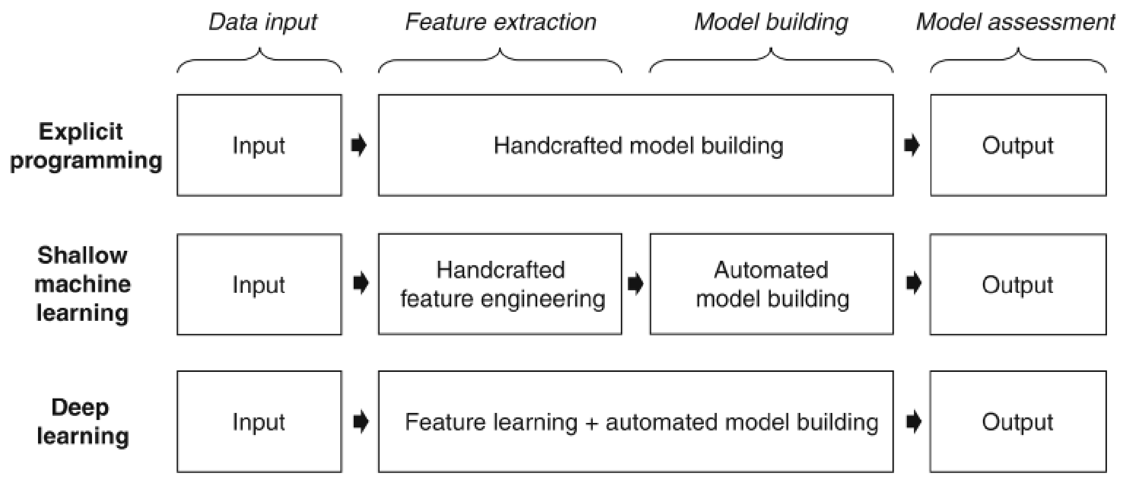


Figure 2 Process of Analytic Model Building (reproduced from [5] *)*

Figure 2 shows the framework on the structure of analytic model building for explicit programming, shallow machine learning and deep learning. The structure of the model is built up by data input, feature extraction, model building and model assessment.

Feature extraction is an important part in model building because of the purpose of the feature extraction is to provide a representation by processing the raw data input. Shallow machine learning very relying on precise features, so its performance is depending on the feature extraction process. With the more development in machine learning, a variety of feature extraction techniques are introduced to process different types of data. For an example, histograms of oriented gradients (HOG), scale-invariant feature transform (SIFT) and the Viola-Jones method are applicable for the analysing the image data. Handcrafted feature engineering is an aggravating job, and it needs a lot of expert knowledges and time to design it which make it time-consuming, labour-intensive and inflexible. The advanced architecture in deep neural networks allows it to perform automated feature learning and extraction from the raw data input with a minimal human intervention. With this capability, DL is better in processing large scale, noisy and unstructured data compared to handcrafted feature engineering. DL has different types of structure and mechanism in feature learning which suitable for different data types and tasks. [5]

For automated model building, the learning algorithm will perform feature extraction from the input data and identify the patterns and relationships which relevant to the learning task. With the capability of automated feature learning, DL can directly operate on high-dimensional raw data input to perform the task of model building. While shallow machine learning and explicit programming are highly depending on the architecture of handcrafted feature engineering to perform the task of model building. [5]

Lastly, the model assessment is to examine the quality of the model from multiple aspects including performance, computational resource and interpretability. Evaluation of the model on the specific task usually can be done with the k-fold cross-validation to prevent underfitting and overfitting. Besides that, the performance of the model can be evaluated by using out-of-sample data that was not included in training samples. [5]

2.2 Transfer Learning

In real life applications, there are a lot of different task and dataset for training and testing. Building a model from scratch for a specific task is a very time consuming and expensive process. Besides that, recollecting the data for training and rebuild the model is extremely difficult or impossible for a new application. Transfer learning is introduced to overcome the problem which can reduce the time and effort to recollect the training data [6]. As an example shown in Figure 3, a regular machine learning requires a training set of cars to learn and predict the cars in the testing set. In the case of transfer learning, the training set can contain images other than car such as bicycle, wheels, etc which similar to the part of the car, or even irrelevant images such as laptop and bird. The irrelevant images seem no connections with the cars, but the images may share the similar edges or patterns with the parts of the car. [7]

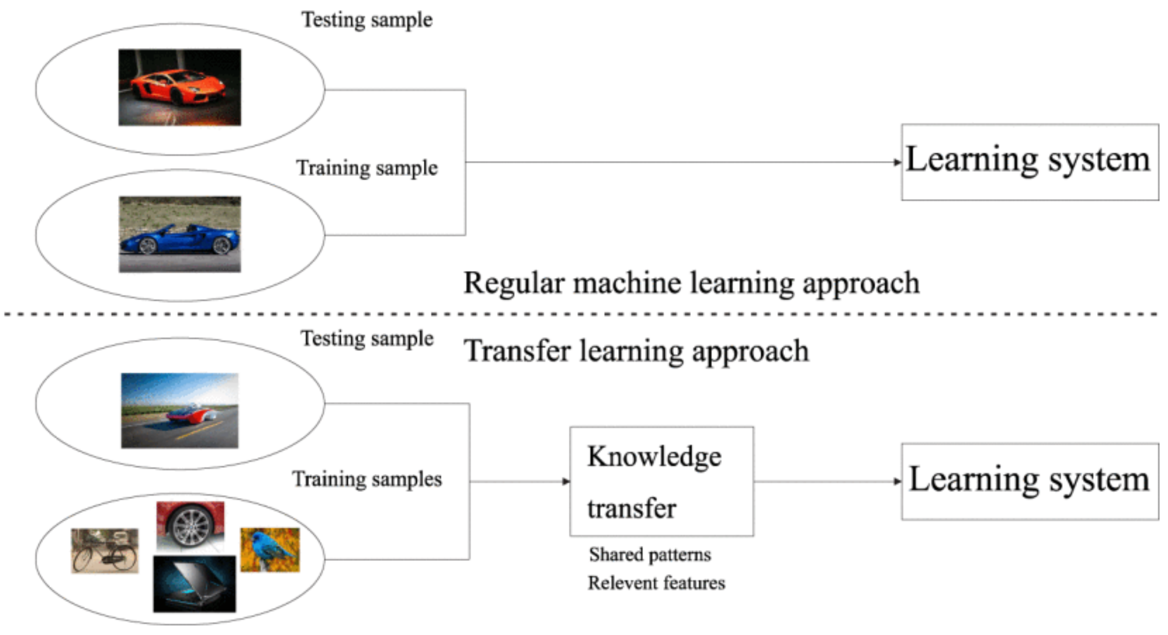


Figure 3 Illustration of Regular Machine Learning and Transfer Learning [7]

2.3 Convolutional Neural Network, CNN

Convolutional Neural Network is a deep learning algorithm which can perform feature extraction and classification. With various combination of datasets, CNN can give a great performance in image classification and recognition. Generally, CNN is built up by five different layers which are input layer, convolution layers, pooling layer, rectified linear unit layer and fully connected layer as shown in Figure 4.[8] Initially, the images are process through the input layer and pass to the convolution layer. Then, the feature of the image data such as gradients, edges, patterns, etc will be extracted and processed in convolution layer which follow by a ReLU activated layer and passed to pooling layer. In pooling layer, the feature maps extracted is simplified by performing maximum or average subsampling of non-overlapping regions in features map to reduce the number of the parameters, resulting decrease the complexity during learning process. Lastly, the fully connected layer will be connected to the pooling layer for classification of the input images. In this layer, flatten or reshape layer is used to reshape the dimension of the feature maps to predicted ouputs then pass to the dense layer with softmax activation for providing the classification outputs.

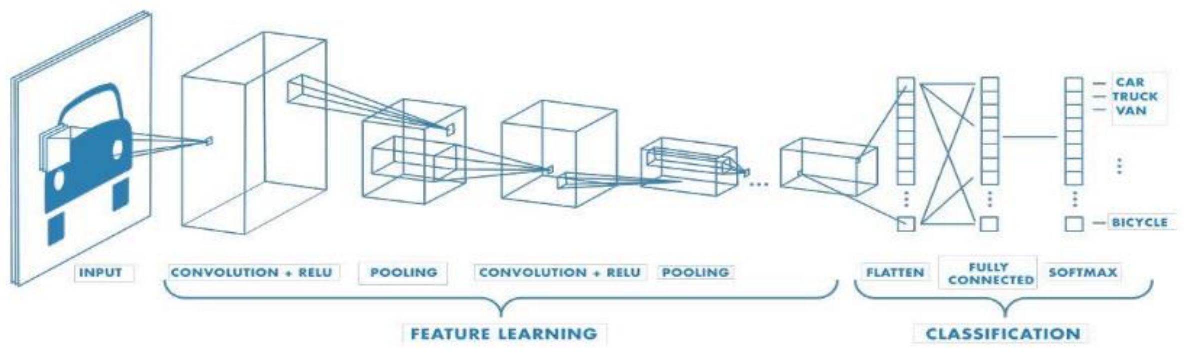


Figure 4 General Architecture of CNN (reproduced from [9] *)*

2.3.1 Input Layer
Input layer is the first layer of a CNN model which receive the images and process them using resize technique for the image can passing into further layer for feature extraction. [9]

2.3.2 Convolution Layer
Convolution layer is the most important layers in CNN. Convolution layer play a role on filtering and extracting the feature of the images, then used as references for calculation of the feature points during evaluation and testing. This layer is followed by a rectified linear unit layer which provide function as mapping negative values to zero and maintaining positive values for a faster and efficient training process. [9]

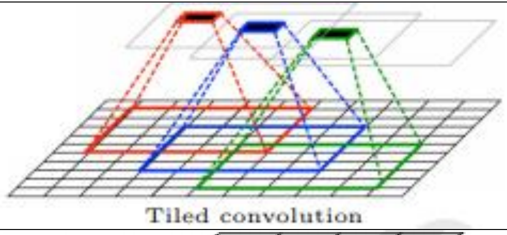
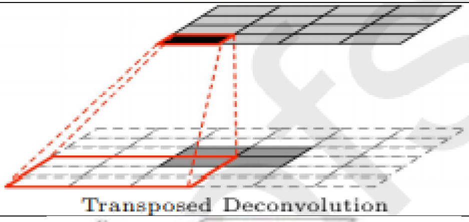
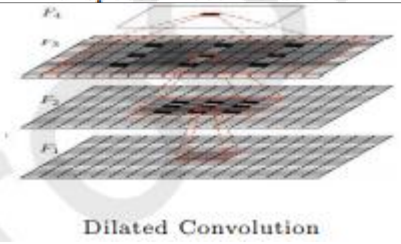
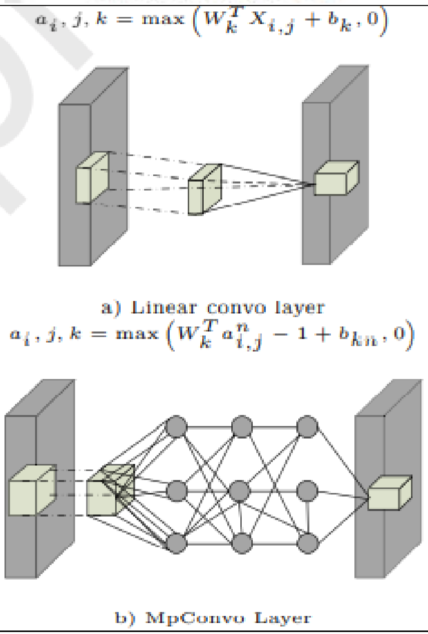
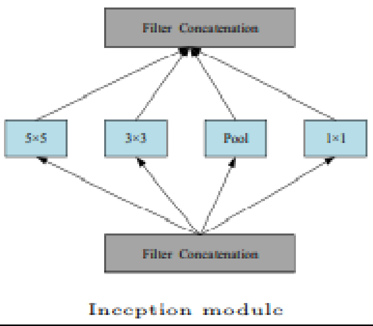
2.3.3 Pooling Layer
Pooling layer receives extracted feature maps from convolution layer and process for simplification for reducing the complexity of the model architecture. Moreover, there are other types of pooling layer such as average pooling, max pooling, global average pooling and global max pooling layer. [9]

2.3.4 Fully Connected Layer
Fully connected layer the last layer of CNN model which provide the output predictions. In this layer, flatten, reshape or global average pooling layer will be used for changing the dimension of the feature map from the convolution and pooling layer. Then, a dense layer with softmax activation will provide the different types of decision according to the learning task given.[9]

2.4 Evolution of CNN
The main core of convolutional neural networks is the convolutional layers and pooling layers. Convolutional layers play important roles in feature learning and extraction which is responsible in most of the computational heavy load. Since from the success of deep neural network AlexNet in 2012, several developments and research are done on CNN for a better performance compared to traditional CNN's. Different types of advancement are done in

convolutional layers such as tiled convolution, transposed convolution, dilated convolution, network in network (NIN) and Inception module as shown in Table 1. [10]

Table 1 Advancement in Convolution (reproduced from [10] *)*

Convolution Method	Visualization
Tiled Convolution	
Transposed Deconvolution	
Dilated Convolution	
Network in Network (NIN)	 $a_{i,j,k} = \max\left(W_k^T X_{i,j} + b_k, 0\right)$ a) Linear convo layer $a_{i,j,k} = \max\left(W_k^T a^n_{i,j} - 1 + b_{kn}, 0\right)$ b) MpConvo Layer
Inception Module	 Filter Concatenation 5×5 3×3 Pool 1×1 Filter Concatenation Inception module

Tiled CNN has the capability to perform learning rotation and scale-invariant features by tilting and multiplying feature maps. The results using tilted CNN show a better performance compared to traditional CNN's. [10]

Transposed convolution can refer as a deconvolution with a reverse process of convolution operation. In this convolution layer, the individual activation is linking to many outgoing activations, while the traditional CNN's is linking different incoming activation to individual activations. [10]

Dilated convolution defined as a convolution layer with a dilated filter. The operation of dilated convolution is similar to a general convolution layer. A same dilated filter at different scales with dilation filters is used in the dilated convolution neural network and the implementation will not affect the structure of the CNN. [11]

NIN can refer to the accumulation of micro-network in CNN. In NIN, non-linear filters or micro networks with composite function had replace the linear filters. This method is to tackle the variance and increase the discrimination power of the local perspective field. [10]

Inception module is introduced by Szegedy et al in CNN by replacing the fixed filter size with variable filter size to provide a more specific and abstract visual features of variable sizes in the network. In Inception module, convolutional layer is placed before the filter and further the dimensionally reduction for increasing the depth of the CNN without increase the complexity of the networks. [10]

2.4.1 Summary of CNN architecture

The evolution of CNN architectures from 1998 to 2019 is shown in Table 2.

Table 2 CNN Architecture Summary (reproduced from [10] *)*

Model	Year of Release	No. of Layers	No. of Parameters	Error Rate: Top 5
Le Net 5	1998	7	60,000	MNIST: 0.9
Alex Net	2012	8	60M	ImageNet: 16.4
VGG16	2014	19	138M	ImageNet: 7.3
Inception V1	2014	22	5M	ImageNet: 6.7
Inception V3	2015	48	24M	ImageNet: 3.5
Inception V4	2016	70	43M	ImageNet: 4.01
ResNet50	2016	50	26M	ImageNet: 5.25
ResNet152	2016	152	25.6M	ImageNet: 3.6
Xception	2016	71	23M	ImageNet: 0.055
FractalNet	2016	20	38.6	CIFAR-10: 7.27
ResNets 101	2015	101	44.6M	ImageNet: 4.60
Inception ResNets	2016	572	56M	ImageNet: 3.52
DenseNet	2017	190	25.6M	CIFAR-10: 5.19
PolyNet	2017	-	92M	ImageNet: 4.25
Resne Xt 50	2017	50	25.5M	ImageNet: 5.96
ResNe Xt 101	2018	101	48.96M	ImageNet: 5.59
Channel Boosted CNN	2018	-	-	-
See Net	2018	152	36.92M	CIFAR-10: 3.58

2.5 Skin Cancer

Cancer is a disease of a cell undergoes uncontrolled and abnormal growth in any part of human body. In severe condition, these cells will spread other parts of the body and destroying them. Skin is the largest organ in human body which can protect us from UV radiation. However, the UV radiation from the sun is becoming more harmful and damaging human due to the depletion of ozone layer increasing rapidly over the last few years. Besides that, people with genetic inclusion, light skin thickness or lack of immunity could get skin cancer.

Seven types of skin cancer which are actinic keratosis (akiec), basal cell carcinoma (bcc), benign keratosis-like lesions (bkl), dermatofibroma (df), melanoma (mel), melanocytic nevi (nv) and vascular lesions (vasc) will used in the model for learning and classification.

Generally, skin cancer can be separate as melanoma skin cancer and non-melanoma skin cancer. Melanoma skin cancer the most dangerous type of skin cancer and had caused most of the death in skin cancer if it is not treated at early stage. Non-melanoma skin cancer includes actinic keratosis(akiec), basal cell carcinoma (bcc), benign keratosis-like lesions (bkl), dermatofibroma (df), melanocytic nevi (nv) and vascular lesions (vasc). Non-melanoma skin cancer usually will not spread to other part of the body which is not life threatening. Non-melanoma skin cancer can be cured through appropriate treatment such as surgery and chemotherapy.

2.5.1 Actinic Keratosis (akiec)

Actinic keratosis is a rough and scaly patch, usually develop on forehead, lips, ears, forearms and scalps or neck of human body. The cause of akiec occurred are long time of exposure under sunlight and indoor equipment for tanning like tanning beds or sunlamps. The UV radiation will damage the the outermost layer of the skin and cause the cells mutate into horn-like growths and bumps with discoloured appearance and rugged or scaly feel. [12]

2.5.2 Basal Cell Carcinoma (bcc)

Basal cell carcinoma is the most common type of skin cancer. Bcc is a skin cancer which develop in the outermost layer of skin and usually does not spread to other tissue or organs. The characteristics of bcc are a small white, or flesh-coloured raised bump, hard, pale white to yellow, shiny red or pink patch which usually grow on the skin. Bcc commonly grow on the neck, the back of the hands and shoulder, but there is a chance that the cancer will appear on any part of the body. [13]

2.5.3 Benign Keratosis-Like Lesions (bkl)

Benign keratosis skin lesions show the features including round or oval in shape, light tan or black colour and flat or slightly raised with a scaly surface. It commonly located at face, chest, shoulders or backs with single or multiple growths.[14]

2.5.4 Dermatofibroma (df)

Dermatofibromas are small, hard, raised skin growths that commonly appear on lower legs, arms or trunks. Dermatofibroma usually developed as a small size of BB pellet with red, pink, purplish, grey or brown colour. [15]

2.5.5 Melanoma (mel)

Melanoma is the most dangerous type of skin cancer which will spread to other tissue or organs, which will lead to dead. Melanoma usually will develop a change in existing mole and a new pigmented or unusual-looking growth on the skin areas that always expose to sun. Moreover, hidden melanoma may appear in areas of human body which little or no exposure to sunlight, such as palms, scalp or genitals which is hard to detect and examinate. [16]

2.5.6 Melanocytic Nevi (nv)

Melanocytic nevi will develop on the skin surface or underneath the skin. The sign and symptoms of Melanocytic nevi are asymmetrical and irregular shape of skin lesion area, variation of color such as brown, tan or pink and evolution of mole.[17]

2.5.7 Vascular Lesions (vasc)

Vascular lesions are small, round and commonly in bloody red colour skin growth that usually found on hands, arms, face, neck, chest and backs. Vascular lesions consist of large number of blood vessels which causing frequent bleeding occurred. [18]

Chapter 3 – Methodology

3.1 Software Objectives

3.1.1 Tensorflow

Tensorflow is an open-source software with which allow user to build a large-scale heterogeneous system for different tasks such as image classification, text classification, face recognition and many more. Moreover, Tensorflow is an excellent tool for implementing experimental approached and productions purposes. [19]

3.1.2 Keras

Keras is an open-source software library written in Python language. Keras is a high-level API which is mainly designed for deep learning and machine learning integrating with machine learning platform, Tensorflow 2. Keras is a powerful tool which can provides different application of models in solving machine learning problems with a focus on deep learning. Besides, Keras models can exported and implement in browser web application and mobile device.

3.1.3 Anaconda

Anaconda is an open-source software that provides toolkit for machine learning, deep learning, data science and many more. Moreover, Anaconda allow users to create environment and download different library software such as Tensorflow, Keras, Numpy and etc in the environment which allow users to code in Python or R. These environments defined as integrated development environments (IDEs) which are a platform or software that convenient and user-friendly for development of code. [20]

3.1.4 Jupyter Notebook

Jupyter notebook is a web-based IDE that allow users to code in their default web browser such as Microsoft Edge, Google Chrome, etc. In Jupyter Notebook, each block of code can be run separately which making it high flexible.

3.1.5 TensorflowJS

TensorflowJS is a famous and mature front-end CNN framework using WebGL-accelerated JavaScript library. The transfer learning applications from Tensorflow integrated Keras library have fully compatible in this framework with both CPU and GPU supports. [21] [22]

3.2 Flow of Model Training

Keras API has a bunch of selection for deep learning model, such as VGG16, Xception, ResNet, Inception, DenseNet, NASNetMobile and many more. These pre-trained models are trained with 14 million with ImageNet dataset.

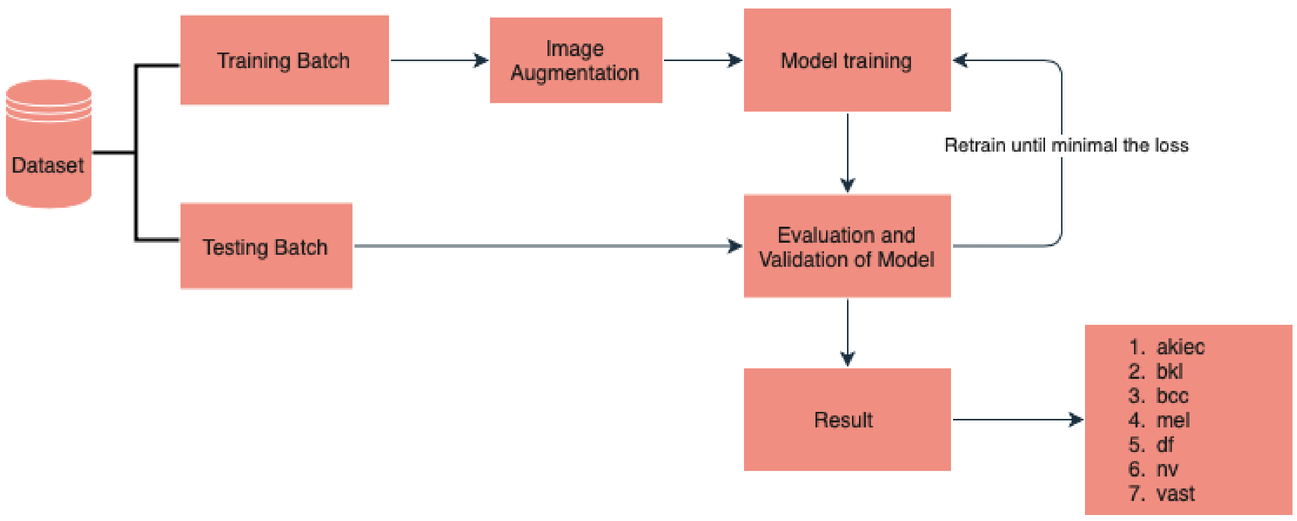


Figure 5 Flowchart of Training Process

In the whole process of training an AI model, dataset is pre-processed by splitting it into training, validation, and testing batches. Next, image augmentation is done if the dataset for different categories is imbalance to prevent overfitting or underfitting. Pre-trained model such as ResNet, DenseNet, etc will undergoes fine-tuning, then will implemented in the training, and fed with the training and validation batches of data which are processed for feature learning and extraction. Finally, the training output will be evaluated with the testing batch of data and retrain until minimal the loss of the model. The process is shown in Figure 5.

3.2.1 Models for Experiment

Due to the limitation of the memory in GPU, CNN architecture below 100MB is chosen to reduce the burden of the GPU and training time. In this paper, four CNN architectures with memory size below 100MB from Keras applications which are Xception, InceptionV3, ResNet50V2 and DenseNet121 are selected and implemented for feature extraction and learning in classification of skin cancers.

In 2015, InceptionV3 which introduced by Google developer had won the 2015 ImageNet Large Scale Visual Recognition Competition (ILSVRC) with top-1 accuracy of 77.9% and top-5 accuracy of 93.7%. InceptionV3 with unique CNN architecture of Inception module is implemented in this paper for observing the performance in classification of skin cancer. [23]

ResNet50V2 is adopted for the skin cancer classification. ResNet50V2 is an improved Residual Network with 50-layer depth model which is taken from Tensorflow2, Keras application compared with ResNet50. Comparing ResNet50V2 with other ResNet model such as ResNet101V2 and ResNet152V2, ResNet50V2 has the smallest size of 98 MB. Due to the deep and heavy weight of ResNet, ResNet50V2 which has the smallest size and less layers is chosen to prevent the overload on the graphic card and the web application. Moreover, ResNet architecture with residual connection which allow more depth developed in the network without early degradation of the accuracy during the training process.[24]

Xception is an improved version from Inception model which introduced in 2016. The CNN architecture of Xception is a combination of depthwise separable convolution and residual connection. Xception model has similar number of parameters with InceptionV3, but the classification performance on ImageNet of Xception is slightly better than InceptionV3. Hence, the partial similar architecture of Xception with InceptionV3 and ResNet is adopted in this paper for comparison of classification performance.

DenseNet121 is a lightweight model with a memory size of 33MB compared to the other three CNNs. Besides that, the depth and parameters of DenseNet121 are 121 and 8M respectively which also smaller than other 3 CNNs as shown in Table 3. Comparing to MobileNet which is same as light-weight model, DenseNet121 is a newer developed model in 2017 with more parameters and depth in the network. DenseNet121 is adopted for classification of skin cancer to investigate the performance between the lightweight CNN and high computational CNN.

3.2.1.1 Residual Network, ResNet

The trend of deep learning over the year is going more deeper to solve more complex and difficult tasks for improve the accuracy of the training model. However, the deeper the neural networks, the training process will become more difficult and leads to saturation and degradation of accuracy. ResNet with a deep residual learning framework is introduced to solve these problems.

From Figure 6, the building block is the main part of deep residual learning framework in ResNet. The identity, x is a “shortcut connection” which allow the outputs to skip the weight layers when the model is trained until optimal, so that the outputs no longer mapping on the weight layers and add neither extra parameters nor computational complexity. As the ResNet can go very deep in convolution layer computing, there is different number of layers in ResNet application such as ResNet101 and ResNet152 which have a total of 101 and 152 layers respectively. Moreover, the computing layers in ResNet can goes deeper up to 1000 layers and more. [24]

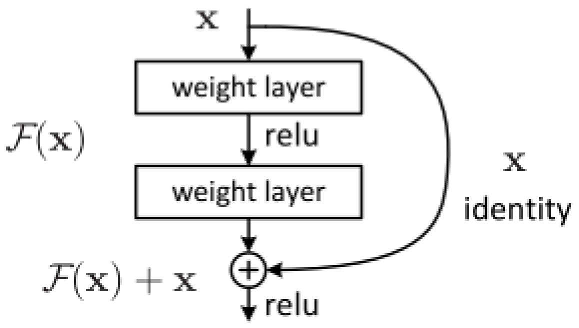

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 6 Building Block of Residual Learning (reproduced from [24] *)*

3.2.1.2 InceptionV3

InceptionV3 is the model using the third generation of Inception module consisting 48 layer of convolution layers which is developed by Google developer. InceptionV3 architecture has module which concatenates the feature maps generated by kernels of varying size as shown in Figure 8. This module is designed to decrease the complexity of the convolution layer by replacing the fixed convolution layer with two or more smaller dimension convolution layer for a more flexible variation. Besides that, the inception module is placed before the filter which aim to increase the computational of the network without increasing the complexity of the architecture. As an example, Figure 7 shows the convolution layer is replaced by one 5x5 convolution, one 3x3 convolution and 1x1 convolution before the filter concat. [2]

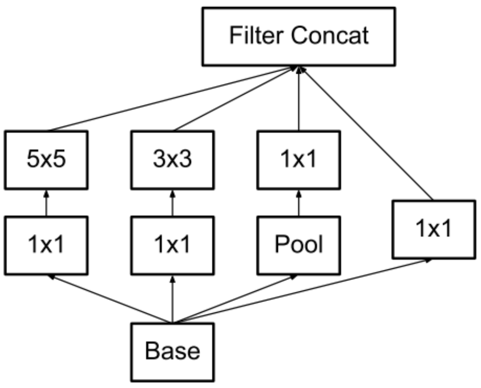


Figure 7 Inception Module in InceptionV3

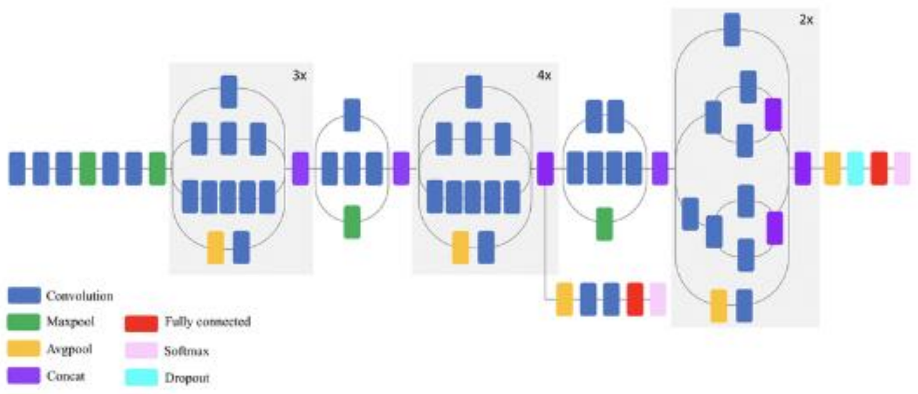


Figure 8 Structure of InceptionV3

3.2.1.3 Xception

Xception model architecture is an improved version form Inception model architecture. The model architecture specification of Xception is shown as the Figure 9.

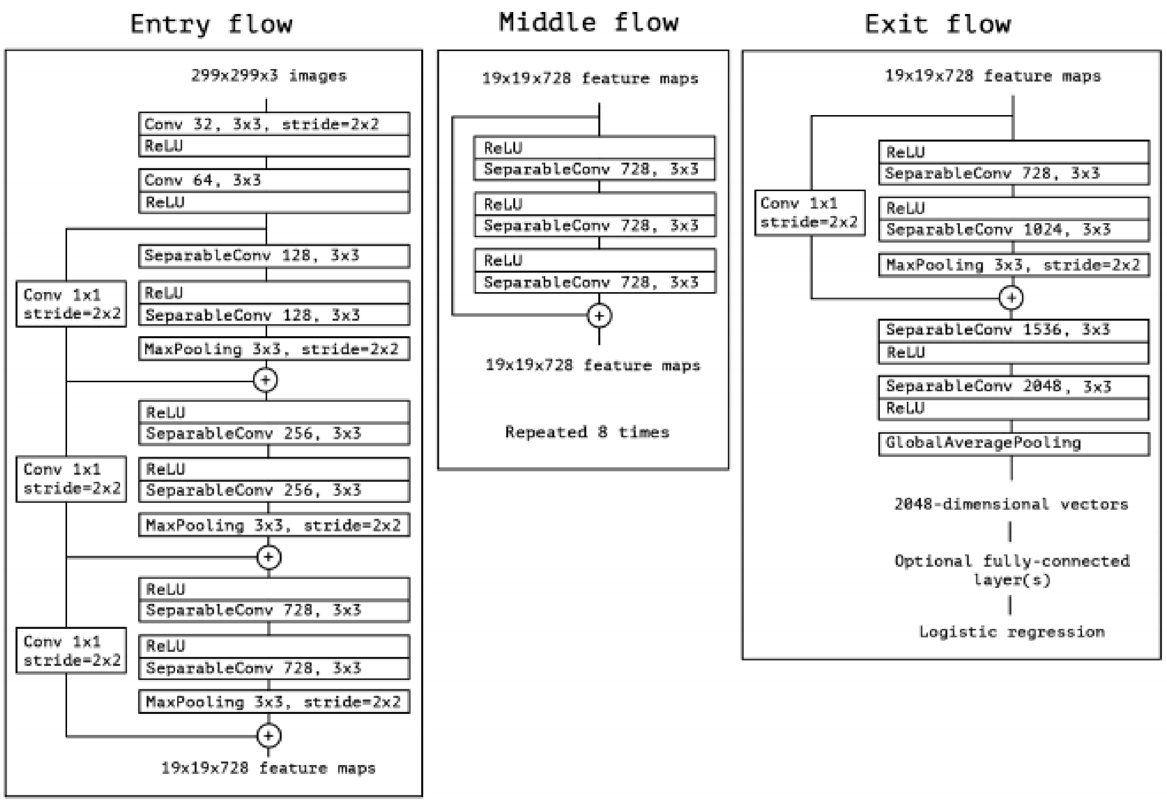


Figure 9 Architecture of Xception

From Figure 9, the architecture of the Xception is a stack of depthwise separable convolution layer connected with residual connection. Xception model has a total of 36 convolution layers which are for feature extraction purpose in the network. Every convolution layer is followed by a ReLU activation logistic regression layer and a batch normalization layer. The input data will first go through the entry flow, then enter to the middle flow for 8 times repeatly and lastly enter the exit flow with fully connected layers.

3.2.1.4 DenseNet121

DenseNet has similar architecture with ResNet. DenseNet has different connectivity pattern between the layers with direct connections from any layer to all subsequent layers as shown in Figure 10. [25] The dense connectivity in DenseNet allows the exchange of the information flow between the layer more efficient and thus decrease the parameters, lighten the vanishing gradient problem and improve the feature map extraction performances. [26]

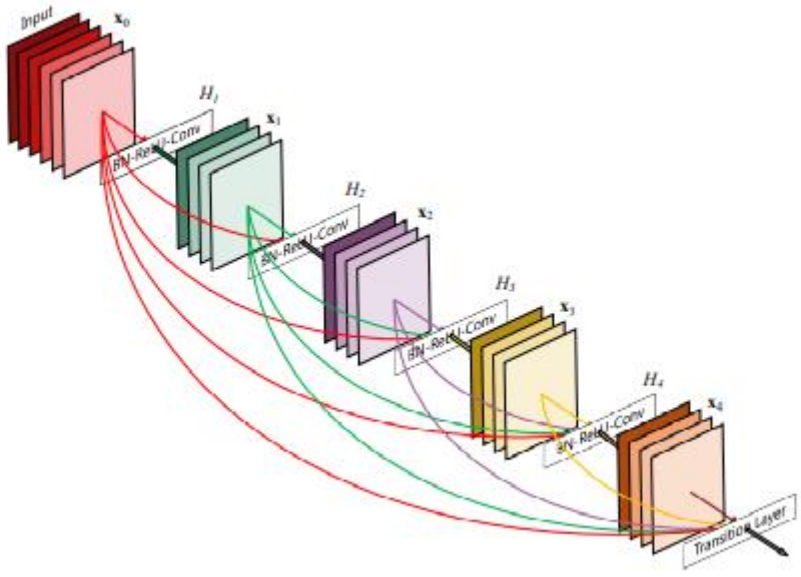


Figure 10 Structure of DenseNet

3.3 Proposed Model Comparison

Table 3 Comparison Table of Proposed Model (reproduced from [27] *)*

Model	Size	Top-1 accuracy	Top-5 accuracy	Parameters	Depth
InceptionV3	92 MB	0.779	0.937	23,851,784	159
Xception	88 MB	0.790	0.945	22,910,480	126
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
DenseNet121	33 MB	0.750	0.923	8,062,504	121

The Table 3 shows the comparison of the four different applications in Keras from 5 different aspects which are size, Top-1 accuracy, Top-5 accuracy, parameters and depth.

InceptionV3, Xception and ResNet50V2 have larger size and parameters compared to DenseNet121. Due to the large parameters, the models will have more computing algorithm and deeper learning for feature extraction and classification.

DenseNet121 is a lightweight model with a slightly poor performance in top-1 and top-5 accuracy compared to the other CNN models. However, DenseNet121 has lesser parameters and complexity in the architecture. The characteristics of DenseNet121 may not put the GPU to overload during the training process.

3.4 Dataset Description

The dataset used in this project is Skin Cancer MNIST: HAM10000 which is taken from Kaggle website. HAM10000 contains a total of 10000 images for 7 different classes of skin cancer diseases. The 7 skin cancers diseases are Bowen’s disease (akiec), basal cell carcinoma (bcc), benign keratosis-like lesions (solar lentigines / seborrheic keratoses and lichen-planus like keratoses, bkl), dermatofibroma (df), melanoma (mel), melanocytic nevi (nv) and vascular lesions (angiomas, angiokeratomas, pyogenic granulomas and hemorrhage, vasc). [3]

3.5 Pre-Process Data

3.5.1 Image Augmentation

Image augmentation is implemented in this study to balance the number of images for 7 classes in dataset. The image is augmented with 7 different operations which are, rotating, width shifting, height shifting, zooming, horizontal flipping, vertical flipping. All the augmented image is resizing to 224x224 pixels. [28]

3.5.2 Batch size

Batch size is one of the most important parameters in deep learning process which can affect the result of the training process. Batch size can be determined as the numbers of the training examples are processed in parallel for training. A larger number of batch size can increase the accuracy of the training accuracy. However, larger number of batch size will consume a lot of computing power which requires a stronger computation hardware such as GPUs.

Due to the current GPU(RTX 2060) has a low memory VRAM of 6 GB, there is a limitation of batch size in this project to prevent long time and resource exhaust of GPU memory in training process.

3.6 Modifying and Training CNNs Network

3.6.1 Proposed Model Design

Fully connected layer at the output of the transfer learning model is modified with adding different types of layer. In the training and evaluation process of CNN, overfitting is a big problem which will affect the accuracy performance of the trained CNN. Dropout layer and batch normalization layer are added to overcome the overfitting problem.

A total of 6 layer which are global average pooling layer, batch normalization layer, dense layer with ReLU avtivation and softmax activation layer and dropout layer are added and fine-tuned. These layers are applied on four different transfer learning model and the proposed layer structure of the models are carried out as shown in Figure 11, 12, 13 and 14.

ResNet50V2

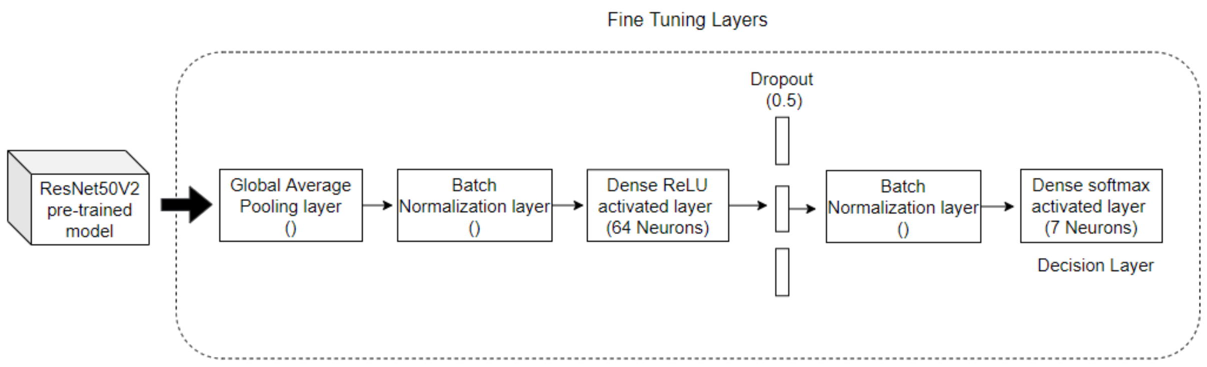

Fine Tuning Layers
ResNet50V2 pre-trained model
Global Average Pooling layer ()
Batch Normalization layer ()
Dense ReLU activated layer (64 Neurons)
Dropout (0.5)
Batch Normalization layer ()
Dense softmax activated layer (7 Neurons)
Decision Layer

Figure 11 Fine Tuned ResNet50V2

InceptionV3

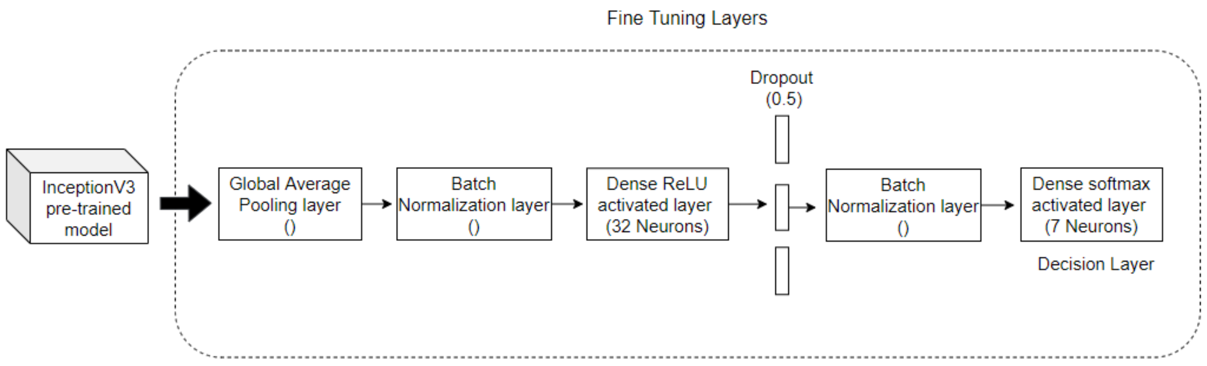

Fine Tuning Layers
InceptionV3 pre-trained model
Global Average Pooling layer ()
Batch Normalization layer ()
Dense ReLU activated layer (32 Neurons)
Dropout (0.5)
Batch Normalization layer ()
Dense softmax activated layer (7 Neurons)
Decision Layer

Figure 12 Fine Tuned InceptionV3

Xception

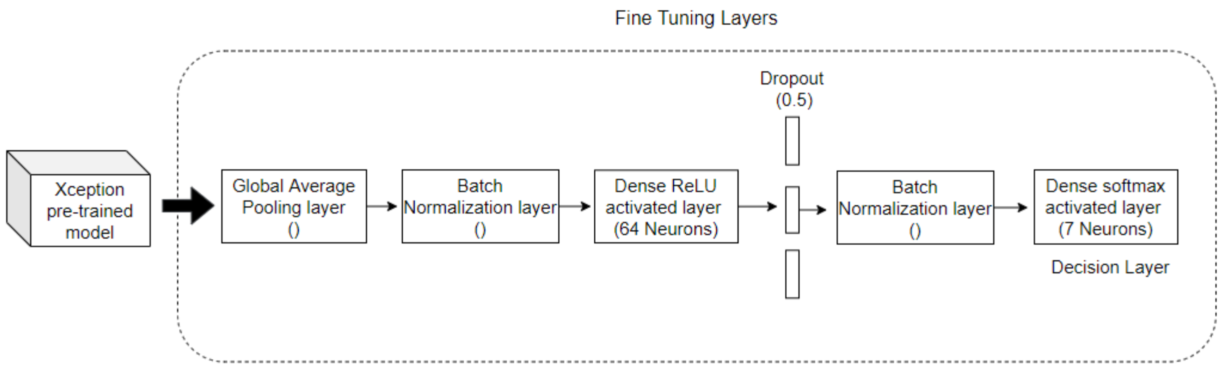

Fine Tuning Layers
Xception pre-trained model
Global Average Pooling layer ()
Batch Normalization layer ()
Dense ReLU activated layer (64 Neurons)
Dropout (0.5)
Batch Normalization layer ()
Dense softmax activated layer (7 Neurons)
Decision Layer

Figure 13 Fine Tuned Xception

DenseNet121

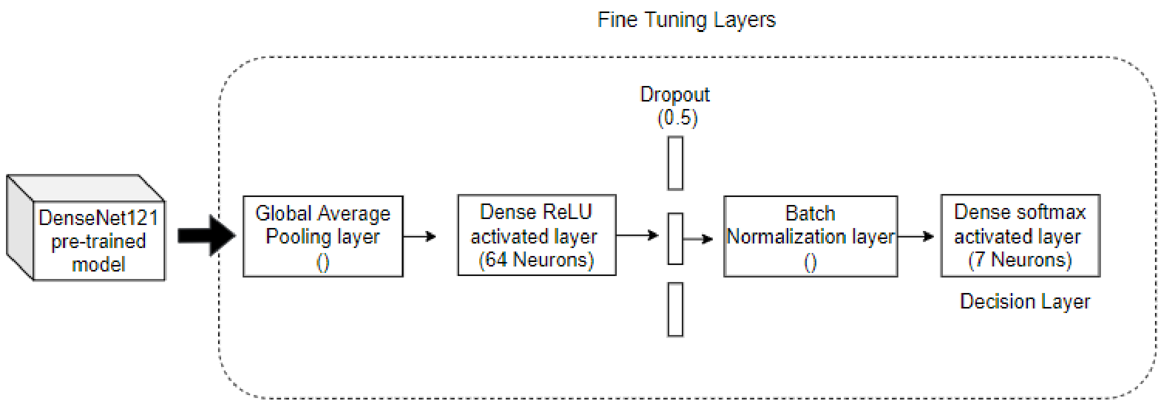

Fine Tuning Layers
DenseNet121 pre-trained model
Global Average Pooling layer ()
Dense ReLU activated layer (64 Neurons)
Dropout (0.5)
Batch Normalization layer ()
Dense softmax activated layer (7 Neurons)
Decision Layer

Figure 14 Fine Tuned DenseNet121

3.6.2 Fine Tuning for Fully Connected Layer

Global Average Pooling Layer

In the first fully connected layer, the weight matrix needs to be compressed before giving the output result. Hence, implementation of layer such as reshape or flatten is required to added to the first fully connected layer, but this will increase the parameters of the network can consumes most of the memory in the network. Global average pooling layer is a type of pooling operation for replace the first fully connected layer which can reduce the memory consumed problems. Global average pooling layer can decrease the number of parameters in the network, resulting in reduction of the complexity of model and minimization of overfitting. [29]

Batch Normalization Layer

Batch Normalization is applied on the neuron activation to maintain the mean output close to 0 and the output standard deviation to 1. Batch normalization can improve the accuracy and training time of the CNN and reduce the training loss in the CNN. [30] In this paper, the batch normalization layer is added after the ReDense layer.

Dropout Layer

Dropout layer is a method to reduce the overfitting during the training of the model. Dropout layer usually is used in the fully connected layers for filtering the output from the neurons in dense layer to decrease the complexity of the architecture. As an example, a dropout layer with a dropout rate of 50%, the coming neural connections to the dropout layer will be decrease 50% during the training process as shown in Figure 15. With the reduction of neural connections, the complexity of the networks will be decrease. [31]

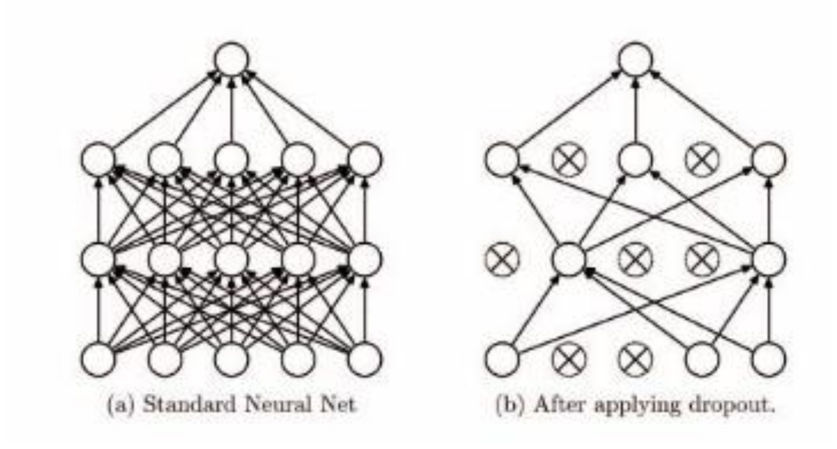


Figure 15 Dropout in Neural Network (reproduced from [31] *)*

ReDense Layer

Dense layer is use in the neural network for creating a fully connection between the neurons. In a Dense layer, every neuron receives input from all neurons of the previous layer, then go through ReLU activation layer for process and pass to next layer, which described as ReDense layer. ReDense layer can improve the performance of several types of CNN with the ReLU activation which can decrease the training loss of the network. Besides that, ReDense layer is a shallow structure which is easy to apply and train without changing the main structure of the network and suffer from local minima or vanishing gradient problem. Figure 16 shows the implementation of ReDense layer in a network and a general network. [32]

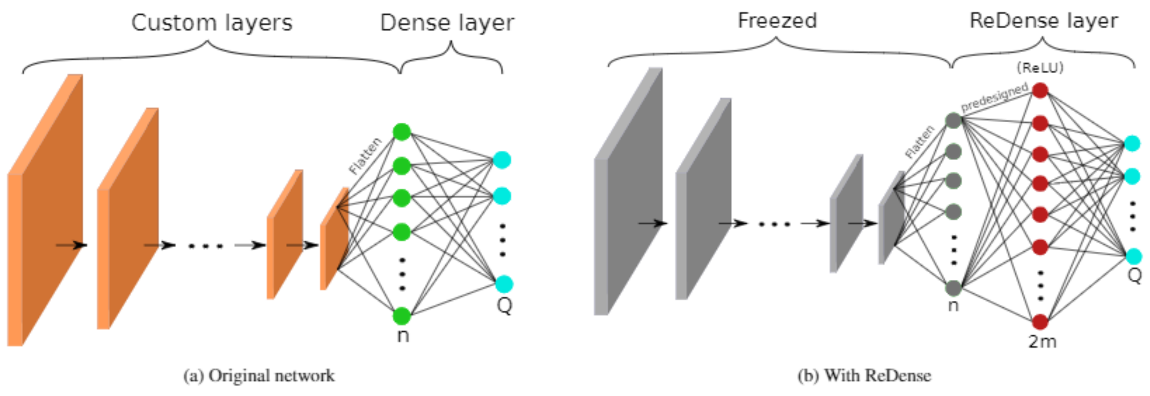


Figure 16 Illustration of Original Network and Network with ReDense (reproduced from [32] *)*

Dense Layer with Softmax Activation

At the last layer of a network, Dense layer with softmax activation is used for providing the output result of the prediction for a N values. The N values are set for each class in the classification task. The function of softmax activation is to normalize the outputs, converting the weight sum values into probabilities for each class in the classification task. [33]

3.6.3 Training Setup

The batch size of image for all models is varies to due to the model. In layer tuning, Different set up is applied to each of the model. Due to the high complexity of InceptionV3 and Xception, the neurons in the first dense layer are set as 32 to decrease the parameters added. For ResNet50V2 and DenseNet121, the neurons are set as 64. Besides that, factor of dropout layer is set as 0.5. An early stop command will be executed when the validation loss of the training process is not improving to prevent the overfitting of the model. Class weight of each category of skin cancer is applied in the training process for a balance accuracy.

3.7 Evaluation of Trained Model

Evaluation and testing results is done for the deep convolutional neural networks and are shown in confusion matrix. 10 % of the dataset which are about 15 images for each class are predicted using the trained model. The result is then compared and justified.

TP (True Positive): The outcomes of prediction on positive is true.

FP (False Positive): The outcomes of prediction on positive is false.

TN (True Negative): The outcomes of prediction on negative is true.

FN (False Negative): The outcomes of prediction on negative is false.

The predicted values are described as positive and negative, and actual values are described as true and false.

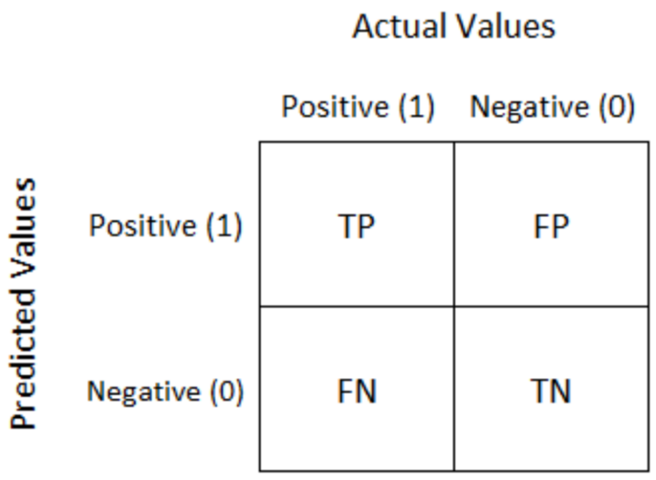


Figure 17 Evaluation of Confusion Matrix

From Figure 17Figure 17, 4 equations are derived as shown at below:

$$Accuracy\ (for\ each\ class) = \frac{TP + TN}{TP + FP + TN + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

$$Sensitivity/Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

[34] [35]

3.8 Implementation of Model in TensorflowJS

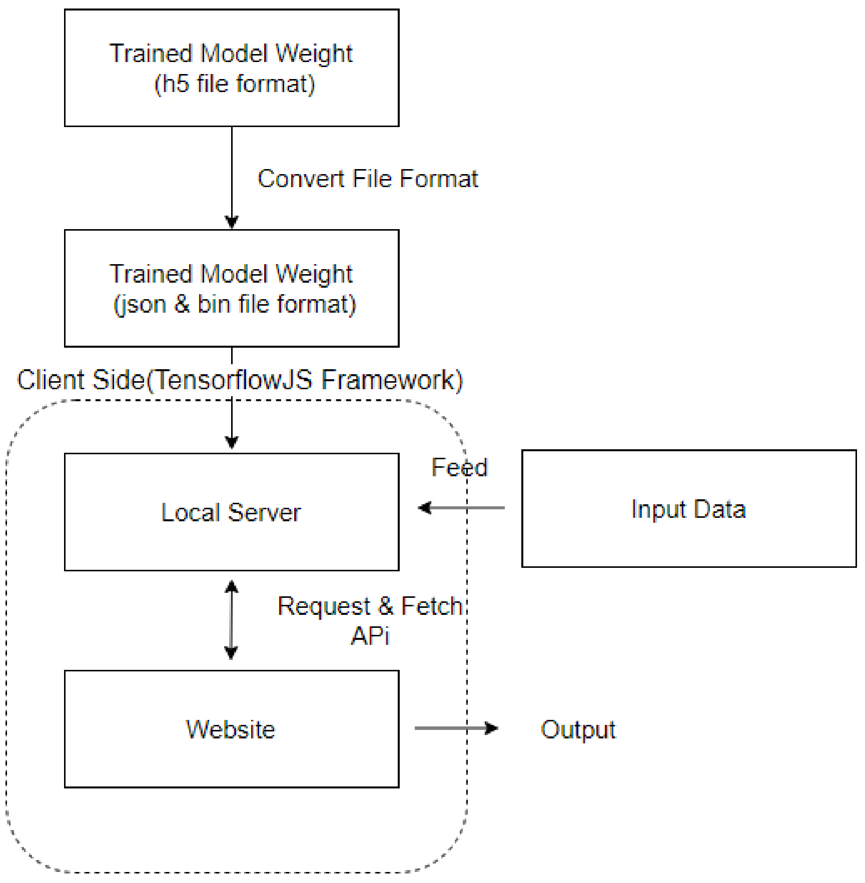

Trained Model Weight (h5 file format)
Convert File Format
Trained Model Weight (json & bin file format)
Client Side(TensorflowJS Framework)
Local Server
Feed
Input Data
Request & Fetch APi
Website
Output

Figure 18 Flowchart of Implementation of TFJS

The trained model weight will be extracted and saved at a specific file location. The file format of model weight will be converted from h5 to json and bin files using TFJS library which are compatible in TensorflowJS framework. The converted file will be saved in the local server of the web application for processing the classification in web application.

Chapter 4 – Result

4.1 InceptionV3

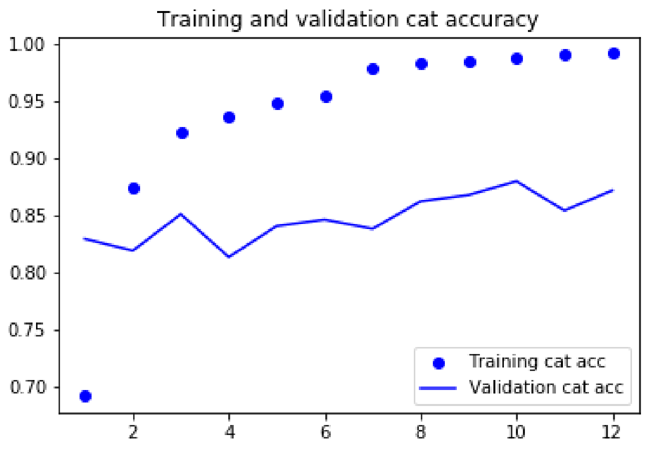


Figure 19 Categorical Accuracy Graph InceptionV3

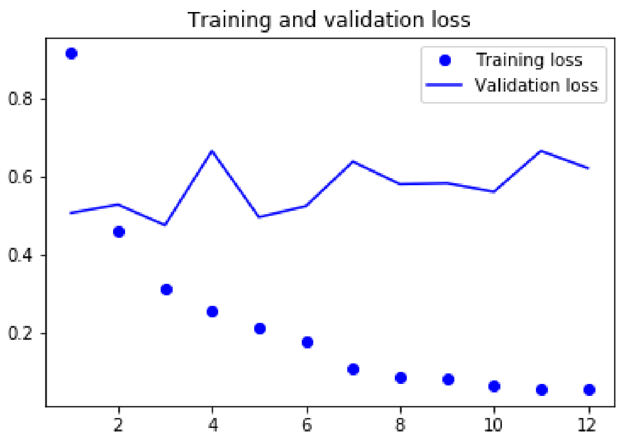


Figure 20 Loss Graph of InceptionV3

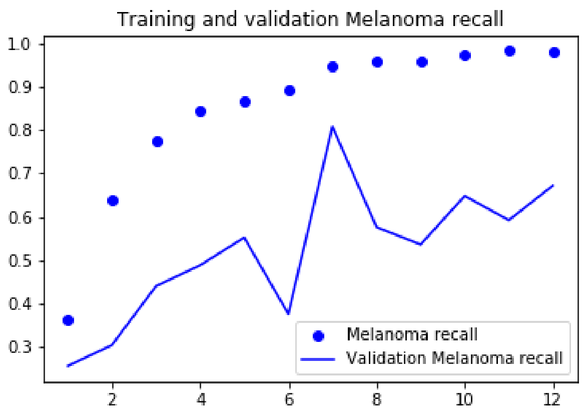


Figure 21 Melanoma Recall Graph of InceptionV3

From Figure 19, the highest validation categorical accuracy is trained at epoch 10 with a value of 0.8797. The validation loss and Melanoma recall at this epoch are 0.5622 and 0.6480 respectively.

Epoch 3 has the lowest validation loss which is 0.4768 as shown in Figure 20. For validation categorical accuracy and Melanoma recall, the results are 0.8510 and 0.4400 respectively.

Figure 21 shows the model trained achieve the highest validation Melanoma recall result of 0.8080 at epoch 7.

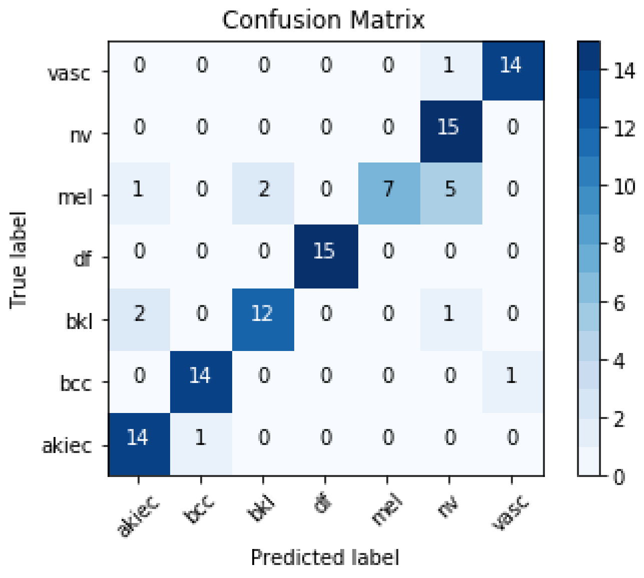
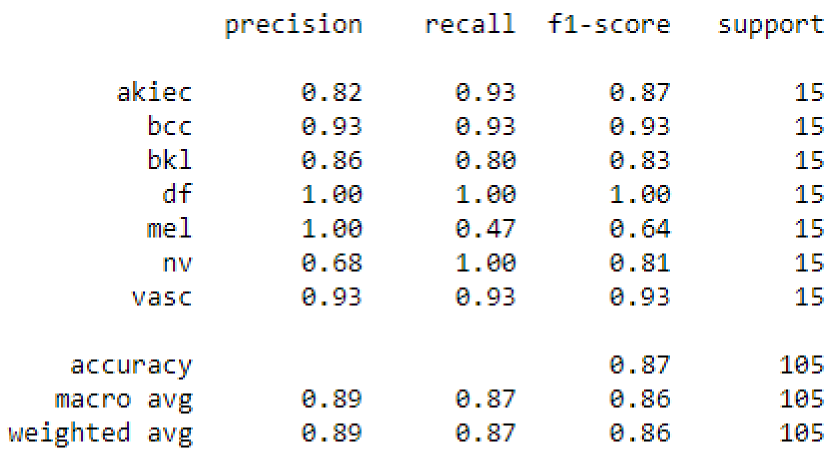


Figure 22 Confusion Matrix of Evaluation, InceptionV3



```
              precision    recall  f1-score   support

       akiec       0.82      0.93      0.87        15
         bcc       0.93      0.93      0.93        15
         bkl       0.86      0.80      0.83        15
          df       1.00      1.00      1.00        15
         mel       1.00      0.47      0.64        15
          nv       0.68      1.00      0.81        15
        vasc       0.93      0.93      0.93        15

    accuracy                           0.87       105
   macro avg       0.89      0.87      0.86       105
weighted avg       0.89      0.87      0.86       105
```

Figure 23 Report of Evaluation, InceptionV3

Figure 22 and Figure 23 show the prediction result of random 105 skin cancer images with 15 images for each category. The overall accuracy is good but still not achieve the goal of 90% above. The confusion matrix shows the accuracy of bkl is slightly poor and mel has the worst performance among the skin cancers. The prediction results for accuracy and recall of mel is unsatisfying with a value of 0.47 compared to other skin cancers. Besides that, bkl has a fairly performance on the accuracy of 0.8. However, the classification results for accuracy on other skin cancers are mainly good which are above 90%.

4.2 Xception

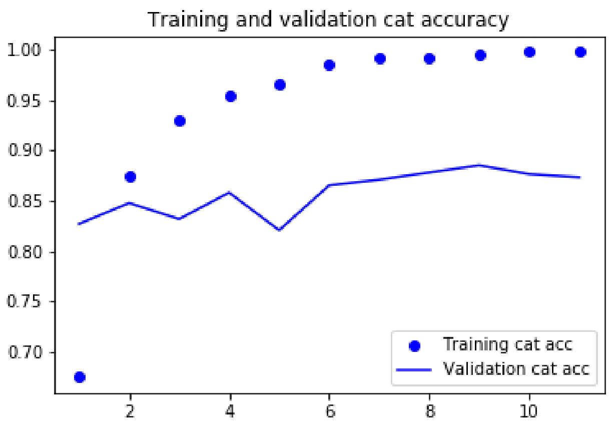


Figure 24 Categorical Accuracy Graph of Xception

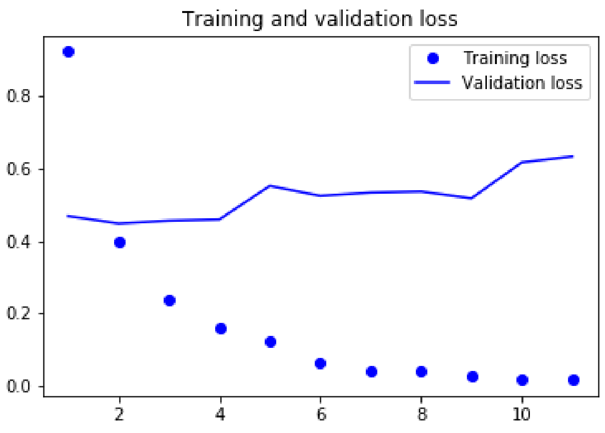


Figure 25 Loss Graph of Xception

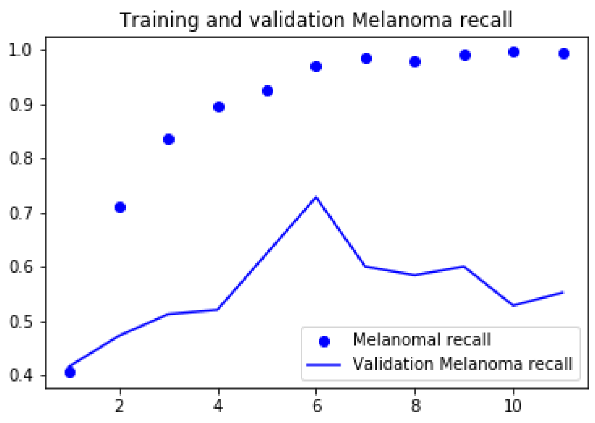


Figure 26 Melanoma Recall Graph of Xception

The lowest validation loss is trained at epoch 2 with a value of 0.4490. The validation categorical accuracy and validation Melanoma recall in this epoch are 0.8478 and 0.4720, respectively.

Epoch 9 have the highest validation categorical accuracy which is 0.8853 with the validation loss and Melanoma recall results are 0.5187 and 0.6000, respectively.

Figure 26 shows the model trained achieve the highest validation Melanoma recall result of 0.7280 at epoch 6.

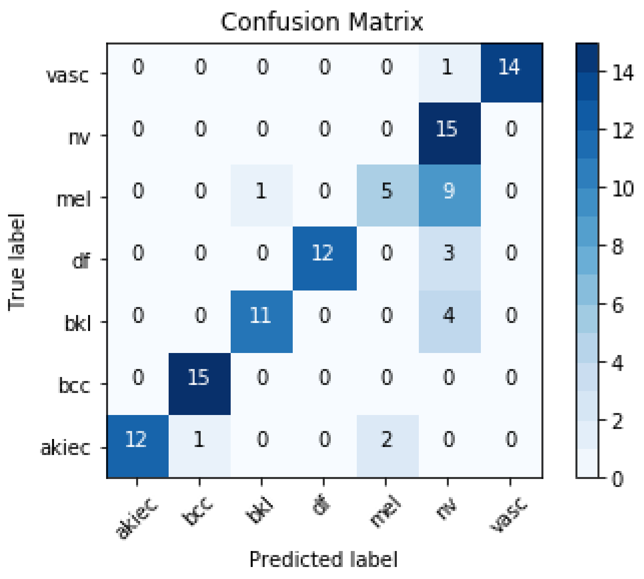
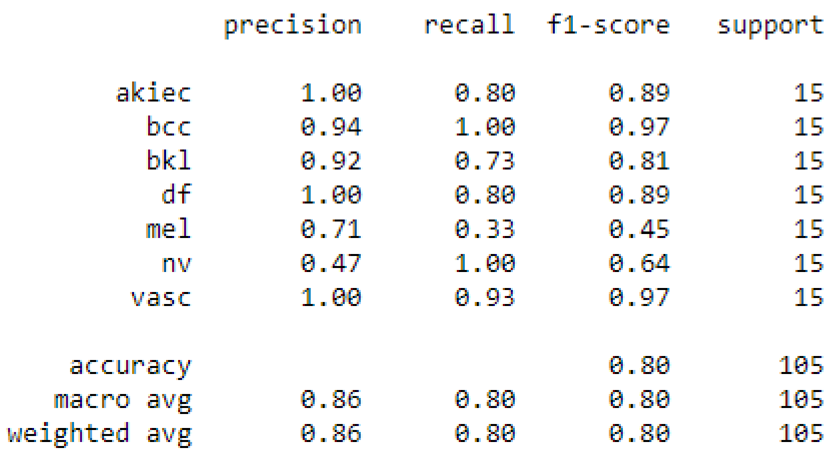


Figure 27 Confusion Matrix of Evaluation, Xception



```
              precision    recall  f1-score   support

       akiec       1.00      0.80      0.89        15
         bcc       0.94      1.00      0.97        15
         bkl       0.92      0.73      0.81        15
          df       1.00      0.80      0.89        15
         mel       0.71      0.33      0.45        15
          nv       0.47      1.00      0.64        15
        vasc       1.00      0.93      0.97        15

    accuracy                           0.80       105
   macro avg       0.86      0.80      0.80       105
weighted avg       0.86      0.80      0.80       105
```

Figure 28 Report of Evaluation, Xception

The overall accuracy of trained Xception model gives a result of 0.80 on the prediction of random 105 skin cancer images with 15 images for each category. The classification results on akiec, df, bkl are slightly poor with values of 0.8, 0.8 and 0.73 respectively, while mel has the worst performances with an accuracy of 0.33.

4.3 ResNet50V2

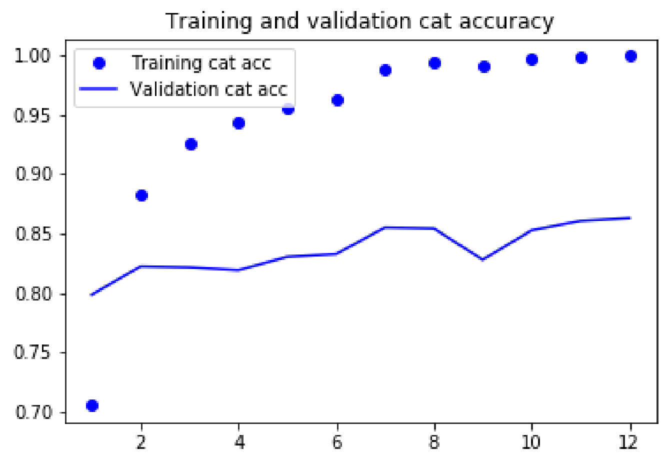


Figure 29 Categorical Accuracy Graph of ResNet50V2

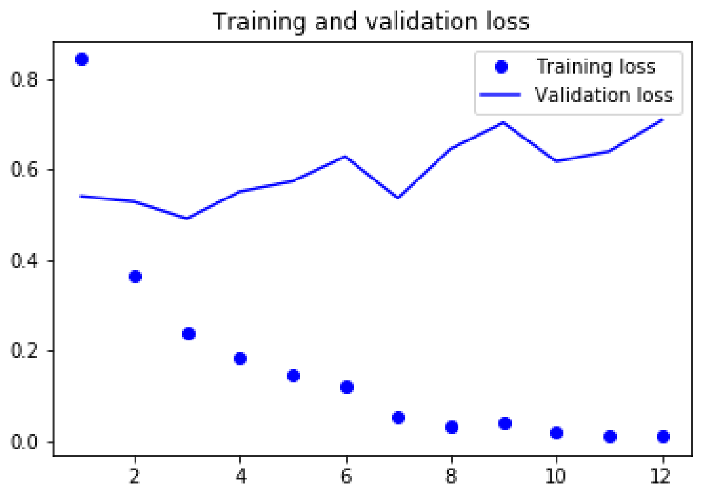


Figure 30 Loss Graph of ResNet50V2

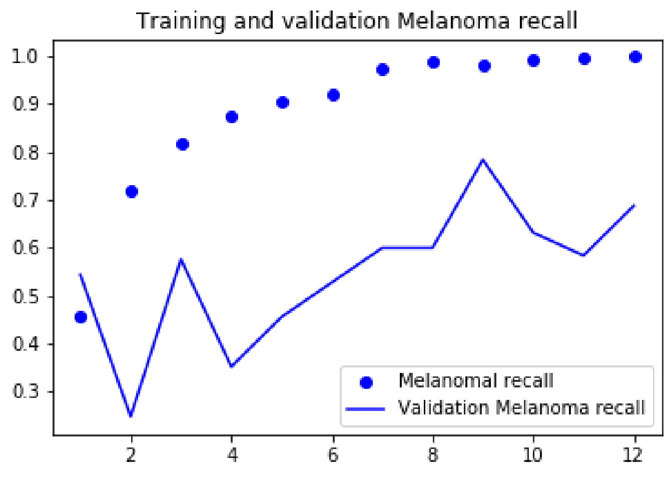


Figure 31 Melanoma Recall Graph of ResNet50V2

Figure 29 shows the highest categorical accuracy of 0.8598 at epoch 12. The validation loss and Melanoma recall at this epoch are 0.6245 and 0.6400 respectively.

The lowest validation loss result is shown in Figure 30 with a value of 0.4432 which is trained on epoch 4. For validation categorical accuracy and Melanoma recall in this epoch, the results are 0.8398 and 0.3760 respectively.

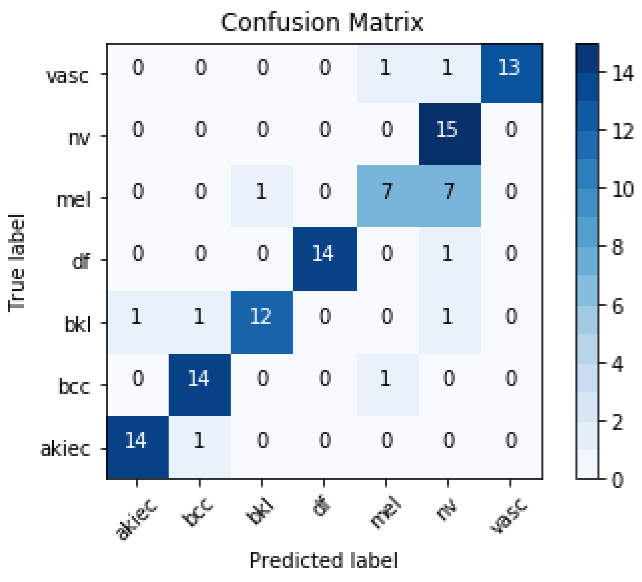


Figure 32 Confusion Matrix of Evaluation, ResNet50V2

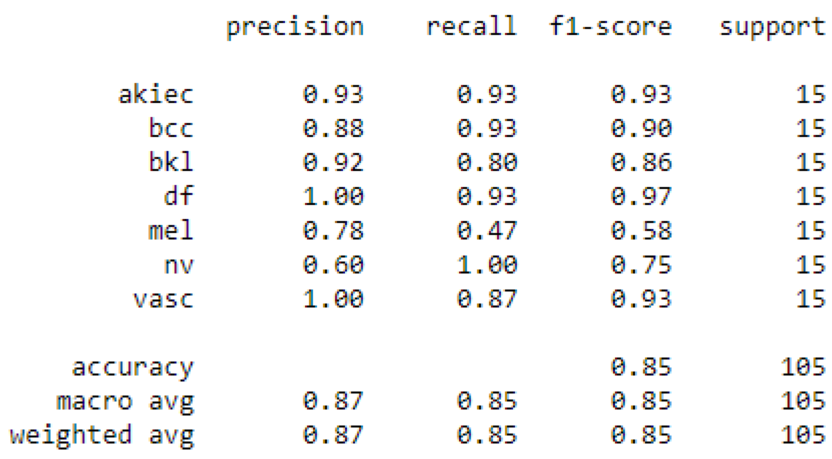

```
              precision    recall  f1-score   support

       akiec       0.93      0.93      0.93        15
         bcc       0.88      0.93      0.90        15
         bkl       0.92      0.80      0.86        15
          df       1.00      0.93      0.97        15
         mel       0.78      0.47      0.58        15
          nv       0.60      1.00      0.75        15
        vasc       1.00      0.87      0.93        15

    accuracy                           0.85       105
   macro avg       0.87      0.85      0.85       105
weighted avg       0.87      0.85      0.85       105
```

Figure 33 Report of Evaluation, ResNet50V2

ResNet50V2 has robust performance in classification of akiec, bcc, df and nv skin cancers with an accuracy above of 90%. The results for bkl and vasc also have fairly performance which are 0.80 and 0.87 respectively. But there is a deficient performance in classification of mel which only has an accuracy of 0.47.

4.4 DenseNet121

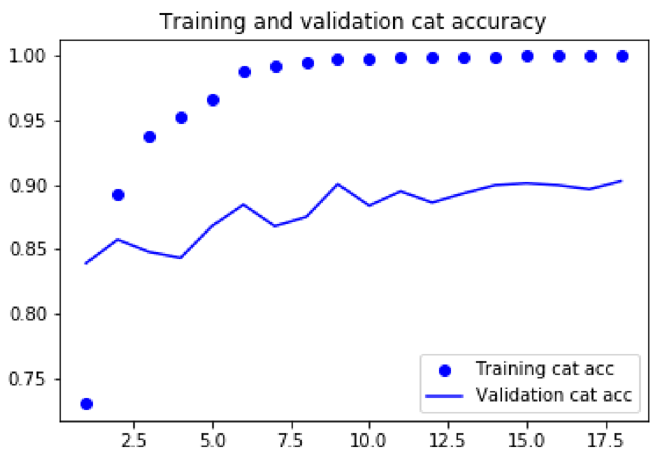


Figure 34 Categorical Accuracy Graph of DenseNet121

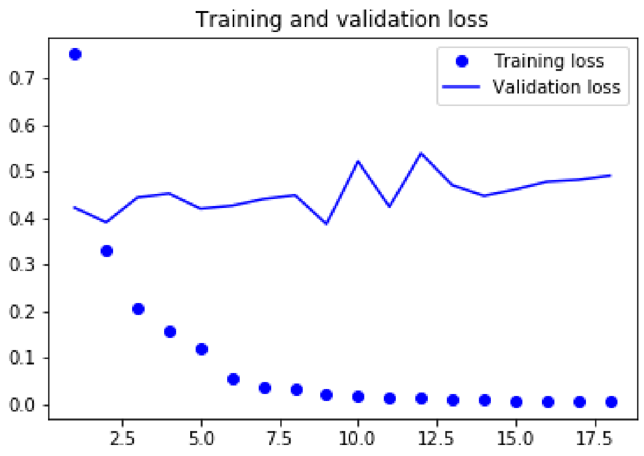


Figure 35 Loss Graph of DenseNet121

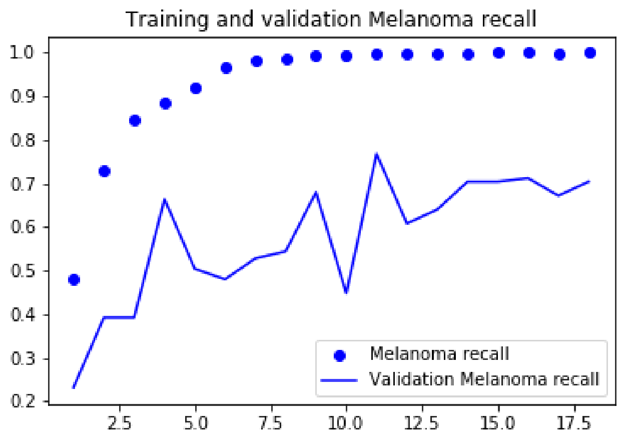
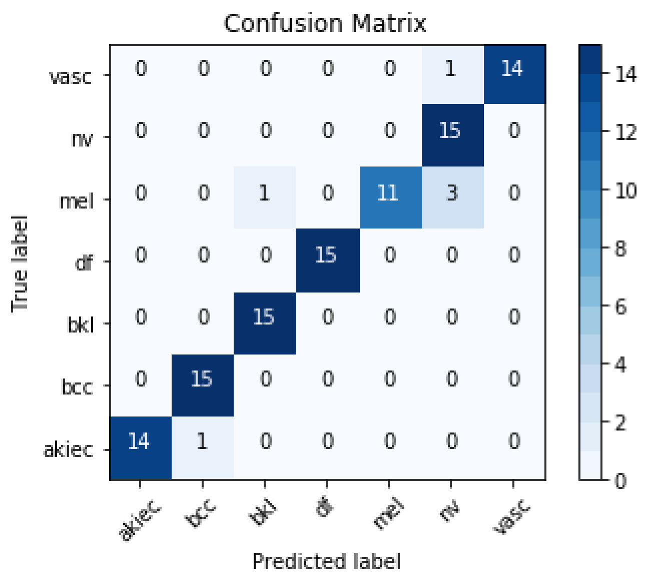


Figure 36 Melanoma Recall Graph of DenseNet121

DenseNet121 training process has stopped at epoch 18 because of no improvement in validation loss. The model achieves the lowest validation loss at epoch 9 as shown in Figure 35. The validation category accuracy, validation loss and validation Melanoma recall in this epoch are 0.9004, 0.3869 and 0.6800 respectively.

Figure 36 shows the highest validation Melanoma recall which is 0.7680 is trained on epoch 11.



True label	akiec	bcc	bkl	df	mel	nv	vasc
vasc	0	0	0	0	0	1	14
nv	0	0	0	0	0	15	0
mel	0	0	1	0	11	3	0
df	0	0	0	15	0	0	0
bkl	0	0	15	0	0	0	0
bcc	0	15	0	0	0	0	0
akiec	14	1	0	0	0	0	0

Figure 37 Confusion Matrix of Evaluation, DenseNet121

```
              precision    recall  f1-score   support

       akiec       1.00      0.93      0.97        15
         bcc       0.94      1.00      0.97        15
         bkl       0.94      1.00      0.97        15
          df       1.00      1.00      1.00        15
         mel       1.00      0.73      0.85        15
          nv       0.79      1.00      0.88        15
        vasc       1.00      0.93      0.97        15

    accuracy                           0.94       105
   macro avg       0.95      0.94      0.94       105
weighted avg       0.95      0.94      0.94       105
```

Figure 38 Report of Evaluation, DenseNet121

Figure 37 and Figure 38 show an impressive performance in the overall accuracy of 0.94 in DenseNet121. But the performance of Melanoma is slightly poor with an accuracy of 0.73 which is the lowest. Except from Melanoma, the other six skin diseases have great result with accuracy above 90%.

4.5 Web Application and BackEnd Classification Results

Table 4 Classification Results in Web Application

105 Skin Cancer Images with 15 Images in Each Category		InceptionV3	Xception	ResNet50V2	DenseNet121
Image Category	Akiec	0.73	0.73	0.67	0.87
	Bcc	0.93	0.93	1.00	1.00
	Bkl	1.00	1.00	1.00	1.00
	Df	1.00	1.00	1.00	1.00
	Mel	0.73	0.73	0.73	0.73
	Nv	1.00	1.00	1.00	1.00
	Vasc	1.00	1.00	0.93	1.00
Average Accuracy		0.92	0.91	0.90	0.94

Table 5 Classification Results in Jupyter Notebook

105 Skin Cancer Images with 15 Images in Each Category		InceptionV3	Xception	ResNet50V2	DenseNet121
Image Category	Akiec	0.93	0.80	0.93	0.93
	Bcc	0.93	1.00	0.93	1.00
	Bkl	0.80	0.73	0.80	1.00
	Df	1.00	0.80	0.93	1.00
	Mel	0.47	0.33	0.47	0.73
	Nv	1.00	1.00	1.00	1.00
	Vasc	0.93	0.93	0.93	0.93
Average Accuracy		0.87	0.80	0.85	0.94

From Table 4, bcc, bkl, df, nv and vasc has particularly satisfactory results in the classification of skin cancer in the web application for all models with an average accuracy of 90% above. However, the results of classification on akiec and mel classes are slightly low compared to other skin cancers. In classification of akiec class, ResNet50V2 performance is slightly poor than other 3 models. While in mel class, all models gave the same classification results.

Table 5 shows the classification results using jupyter notebook. In Xception model, the classification results of mel and bkl classes are lower than other three models with accuracy of 0.33 and 0.73 respectively. Thus, Xception models has a lower average accuracy of 0.8. InceptionV3 and ResNet50V2 have similar average accuracy with a different of 0.02.

By observing Table 4 and Table 5, DenseNet121 has outperformed the other models in prediction of all categories of skin cancer with an average accuracy of 0.94.

4.6 Justification of Training Results

Table 6 Comparison of Best Epoch for Each Model

	InceptionV3	Xception	ResNet50V2	DenseNet121
Best Epoch	3	2	4	9
Validation Categorical Accuracy	0.8510	0.8478	0.8398	0.9004
Validation Loss	0.4768	0.4490	0.4432	0.3869
Validation Melanoma Recall	0.4400	0.4720	0.3760	0.6800

Table 6 shows the training result of lowest loss epoch for each model including the accuracy, loss and Melanoma recall. DenseNet121 has the best performance in three results which are validation categorical accuracy, loss and Melanoma recall. Other three models also have good performance in accuracy, but the validation loss and Melanoma recall results are slightly worse compared to DenseNet121.

InceptionV3, Xception and ResNet50V2 have larger size and parameters which causing the complexity of the models higher than DenseNet121. Thus, the learning weights in these models are easily saturated at the initial of the training and causing higher loss in the following epochs until the early stop of training command execute.

DenseNet121 is a lightweight model with a smaller size and parameters which is suitable to implement in mobile application. The low complexity of the model allows the training process undergoes more feature learning and achieve the lowest validation loss among these models.

Besides that, same layer of fine-tuning is applied to all the models. Some of the models may need some different layer tuning for a better performance in training due to their unique architecture.

Due to the early saturation of the training process, the validation Melanoma recall results are not satisfying. From Figure 21, Figure 26, Figure 31 and Figure 36, the results of validation Melanoma recall perform well after trained for a few epochs which is at a range between epoch 6 and 12. However, the losses in these epochs are too high and unacceptable which may affect the accuracy of the classification.

4.7 Justification of Classification Results

Table 7 Prediction Result with Random 105 Skin Cancer Images

Prediction Platform		InceptionV3	Xception	ResNet50V2	DenseNet121
Jupyter Notebook	Overall Accuracy	0.87	0.80	0.85	0.94
Web Application	Overall Accuracy	0.92	0.91	0.90	0.94
Differences		0.05	0.11	0.05	0

Classification accuracy of DenseNet121 on backend using jupyter notebook are fully match with the results in web application. Besides that, there is a different of 5% between the average accuracy of backend and the web application in the classification results of InceptionV3 and ResNet50V2 models which still acceptable. However, the different of average accuracy in Xception model for backend and web application had reach 10% which is unacceptable and unreliable.

Xception, InceptionV3 and ResNet50V2 have more complex architecture and parameters compared to DenseNet121. The complexity of the models requires to consume plenty of memory in GPU for learning, training and classification process. Hence, the memory in GPU is allocated continuously during the python code is executing, resulting the memory of the GPU is not able to be released back to the system and causes the memory resources exhausted errors occurred. When the memory resource is not sufficient to continue process the input data, the classification output of the model may less computing power to process it and provide an unreliable result.

Chapter 5 – Conclusion

5.1 Conclusion

This paper analysed the classification accuracy of four different deep convolutional neural networks (D-CNN) on skin cancer MNIST: HAM10000 from Kaggle website. Transfer learning is adopted with using four different applications in Keras which are InceptionV3, Xception, ResNet50V2 and DenseNet121.The main goal is to train the models with the skin cancer images and achieve an accuracy of 90% above with lowest loss and evaluating the performance of the classification of models in both web application and jupyter notebook. The results show a significant performance of classification in web application for four models with an average accuracy of 90% above. However, the classification performance in jupyter notebook for InceptionV3, Xception and ResNet50V2 are unsatisfying which are between 80% and 87% average accuracy of 0.94. Besides that, there is a different of 5%, 5% and 11% in the average accuracy of classification performance for InceptionV3, ResNet50V2 and Xception respectively between web application and jupyter notebook, resulting the output prediction of the models may not reliable.

5.2 Deliverables

1. A complete cloud-based web application for skin cancer classification on image uploaded by users.

The deliverable is completed, as refer to Figure 46.

5.3 Reflection on Timeline and Project

At initial of the project, I had focused more research and familiar on computer science areas which is development of mobile application using JavaScript. Besides that, I had done the classification of skin cancer with the trained model weight provided by past FYP student using Jupyter Notebook. Then, I had presented my progress and work in the first moderation to Dr. Ananda, and he had pointed out my research is too focus on computer science. Dr. Ananda also gave me some advice to focus more research on deep learning algorithm and training process. Moreover, I had discussed with my supervisor, Dr Hermawan and concluded to train a CNN which has more computational ability compared to lightweight model.

After the first moderation and discussion with supervisor, I had changed my gantt chart planning by started to learn and train a CNN for a better accuracy and lower loss compared to past FYP students instead of continue developing mobile application. Due to the size of CNN trained has larger size and parameters, mobile application is not suitable platform for implementation of the CNN. Hence, I had to develop web application for the CNN implementation again. I found out that the training process of the CNN and development of web application are time consuming process which I had to delay my plan of writing thesis from March to April.

For training the CNN, I had faced a lot of challenges in learning to use Tensorflow, Keras libraries and Jupyter Notebook. However, I managed to build and train the CNN for a desired result by referring to journal papers, youtube, google searching and coding forum website such as Github and StackOverFlow. Although the time taken is exceed my planning in gantt chart, I had managed to complete the web application with implementation of CNN.

For the web application, I had facing a challenge when using Flask APIs that the classification output of the web application always remains the same. The performance and results are unsatisfying and unacceptable. After trying debug, no solution can solve the problems because the exhaust resource error in the memory of GPU had limit the computing process of the CNN except changing the hardware, GPU. Hence, I had changed the APIs from Flask to TensorflowJS. I used a template of TensoflowJS framework which is compatible with Keras applications from GitHub and done some fine-tuning in the template such as model conversion and download and configuration of classification output. The web application using TensorflowJS framework can function well by providing classification results with the skin cancer images uploaded.

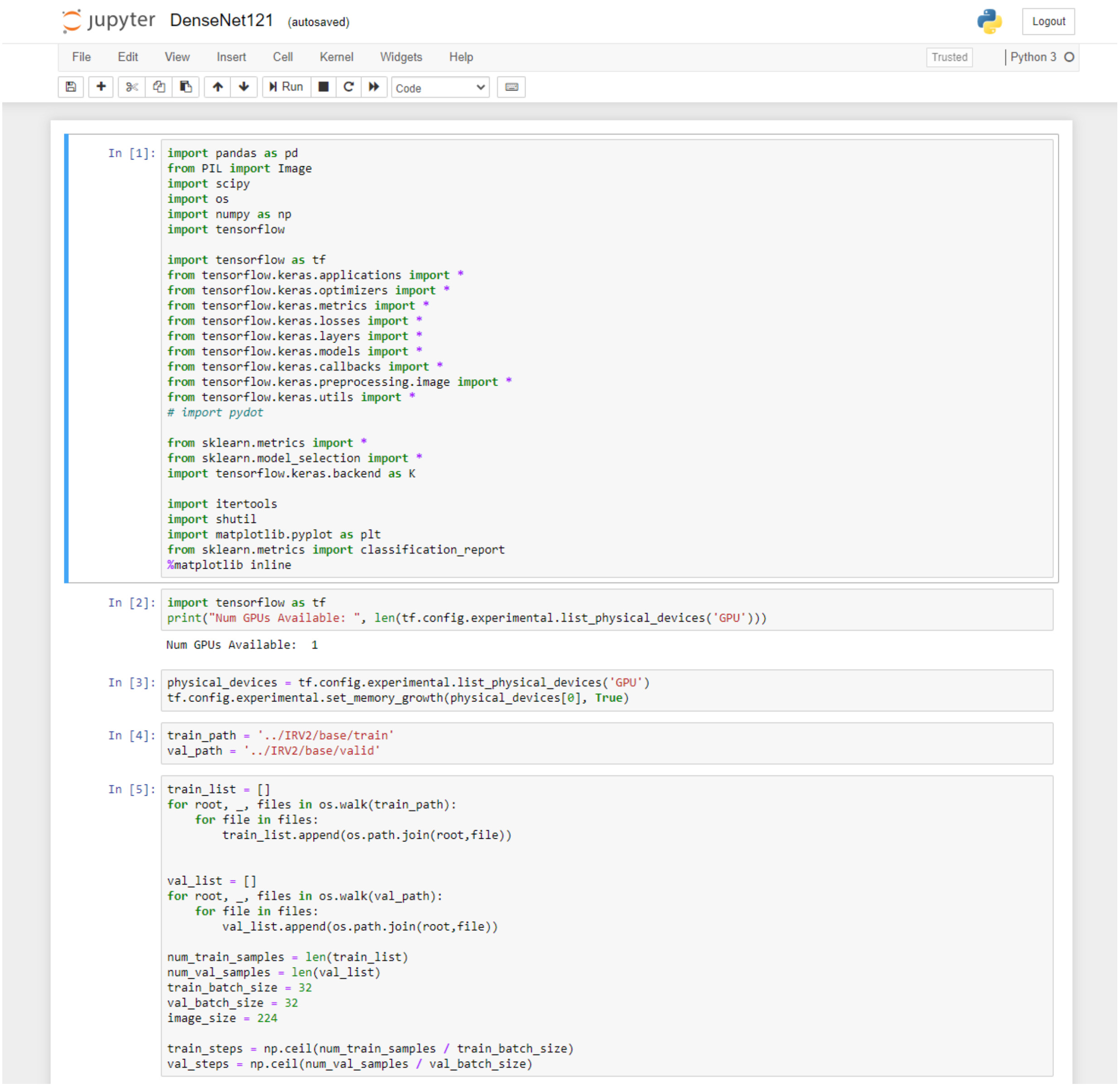
Lastly, I had done the main goal of the web application before April and start writing my thesis. The planning of writing thesis is delayed, but I had managed to complete before the submission deadline.

5.4 Future Works

DenseNet121 had outperformed the other three trained models by remaining the same accuracy in both web application and jupyter notebook. DenseNet is well tuned and trained for a valid classification performance and in both web application and jupyter notebook without any differences or errors.

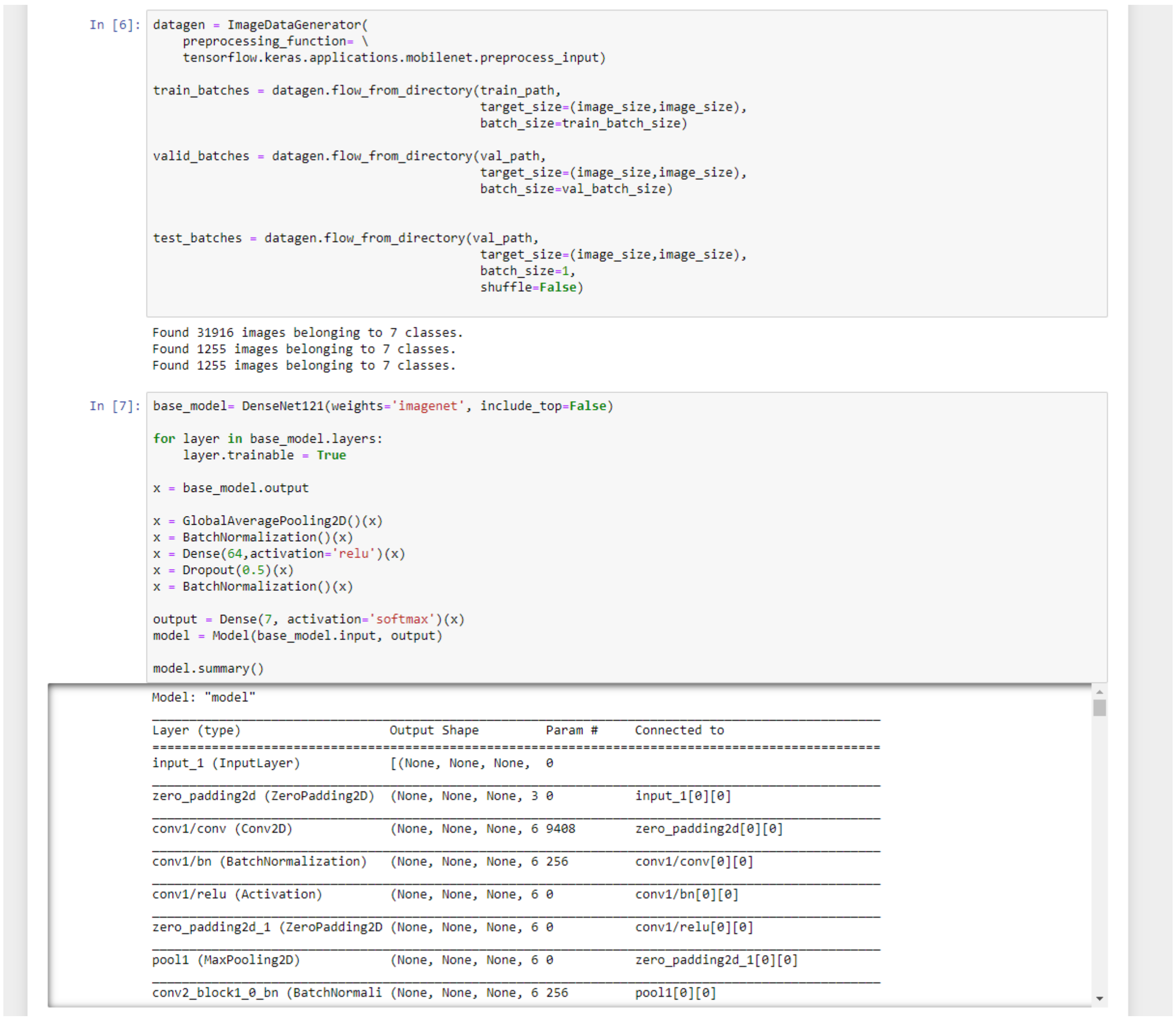
There still a lot of improvement can be done for the models training and web implementation. In future, implementation of stronger GPU with more memory and strong computing power enables the layer and parameters tuning can go for more complex and higher for a better training and learning process. Moreover, the batch size of the dataset can be increase for a higher value without worry about the resource exhaust error in the GPU.

Appendix


jupyter DenseNet121 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [1]: import pandas as pd
        from PIL import Image
        import scipy
        import os
        import numpy as np
        import tensorflow

        import tensorflow as tf
        from tensorflow.keras.applications import *
        from tensorflow.keras.optimizers import *
        from tensorflow.keras.metrics import *
        from tensorflow.keras.losses import *
        from tensorflow.keras.layers import *
        from tensorflow.keras.models import *
        from tensorflow.keras.callbacks import *
        from tensorflow.keras.preprocessing.image import *
        from tensorflow.keras.utils import *
        # import pydot

        from sklearn.metrics import *
        from sklearn.model_selection import *
        import tensorflow.keras.backend as K

        import itertools
        import shutil
        import matplotlib.pyplot as plt
        from sklearn.metrics import classification_report
        %matplotlib inline

In [2]: import tensorflow as tf
        print("Num GPUs Available: ", len(tf.config.experimental.list_physical_devices('GPU')))

        Num GPUs Available:  1

In [3]: physical_devices = tf.config.experimental.list_physical_devices('GPU')
        tf.config.experimental.set_memory_growth(physical_devices[0], True)

In [4]: train_path = '../IRV2/base/train'
        val_path = '../IRV2/base/valid'

In [5]: train_list = []
        for root, _, files in os.walk(train_path):
            for file in files:
                train_list.append(os.path.join(root,file))


        val_list = []
        for root, _, files in os.walk(val_path):
            for file in files:
                val_list.append(os.path.join(root,file))

        num_train_samples = len(train_list)
        num_val_samples = len(val_list)
        train_batch_size = 32
        val_batch_size = 32
        image_size = 224

        train_steps = np.ceil(num_train_samples / train_batch_size)
        val_steps = np.ceil(num_val_samples / val_batch_size)
```

Figure 39 Pre-Process of Data



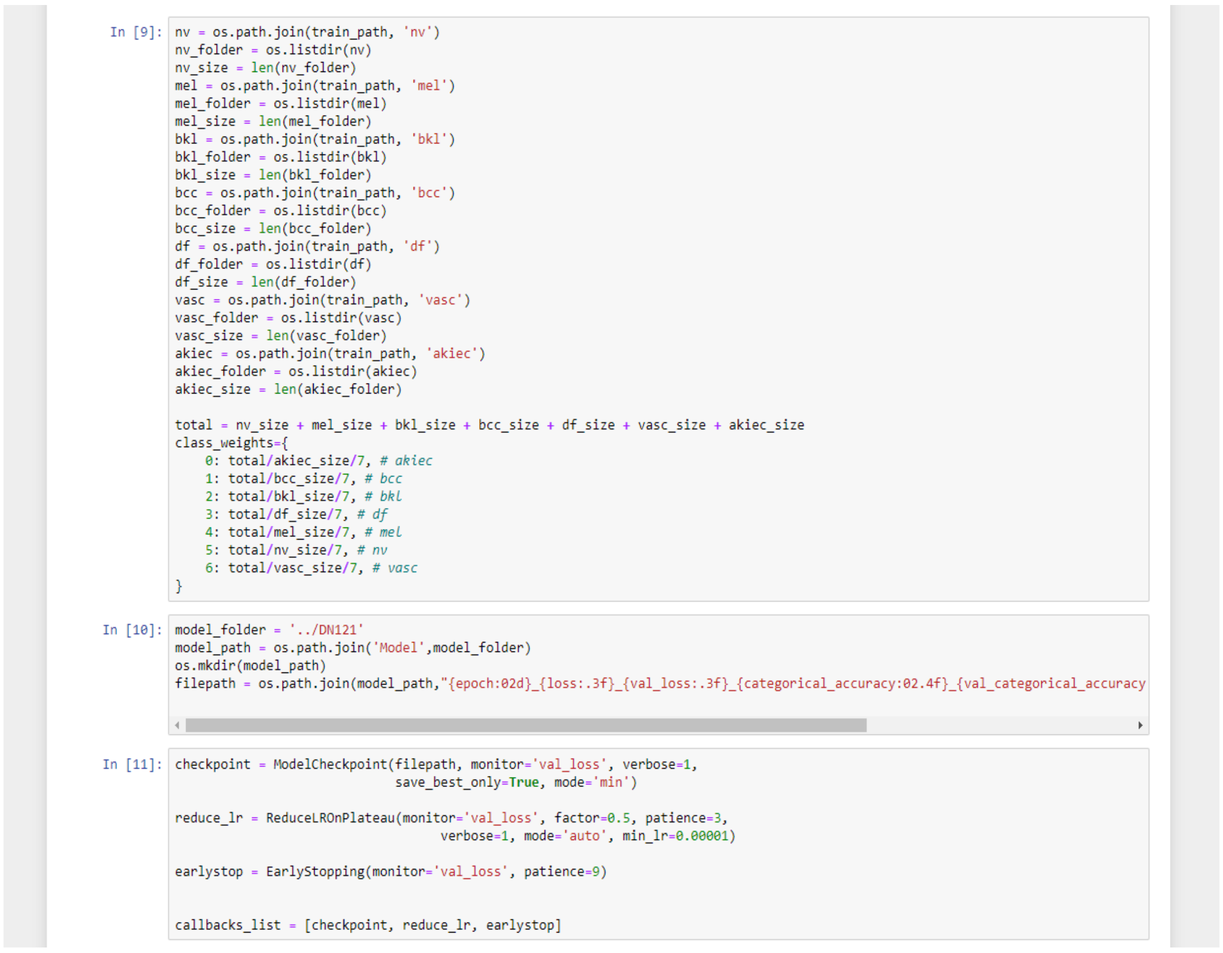
```
In [6]: datagen = ImageDataGenerator(
            preprocessing_function= \
            tensorflow.keras.applications.mobilenet.preprocess_input)

        train_batches = datagen.flow_from_directory(train_path,
                                                    target_size=(image_size,image_size),
                                                    batch_size=train_batch_size)

        valid_batches = datagen.flow_from_directory(val_path,
                                                    target_size=(image_size,image_size),
                                                    batch_size=val_batch_size)


        test_batches = datagen.flow_from_directory(val_path,
                                                    target_size=(image_size,image_size),
                                                    batch_size=1,
                                                    shuffle=False)
```

```
Found 31916 images belonging to 7 classes.
Found 1255 images belonging to 7 classes.
Found 1255 images belonging to 7 classes.
```

```
In [7]: base_model= DenseNet121(weights='imagenet', include_top=False)

        for layer in base_model.layers:
            layer.trainable = True

        x = base_model.output

        x = GlobalAveragePooling2D()(x)
        x = BatchNormalization()(x)
        x = Dense(64,activation='relu')(x)
        x = Dropout(0.5)(x)
        x = BatchNormalization()(x)

        output = Dense(7, activation='softmax')(x)
        model = Model(base_model.input, output)

        model.summary()
```

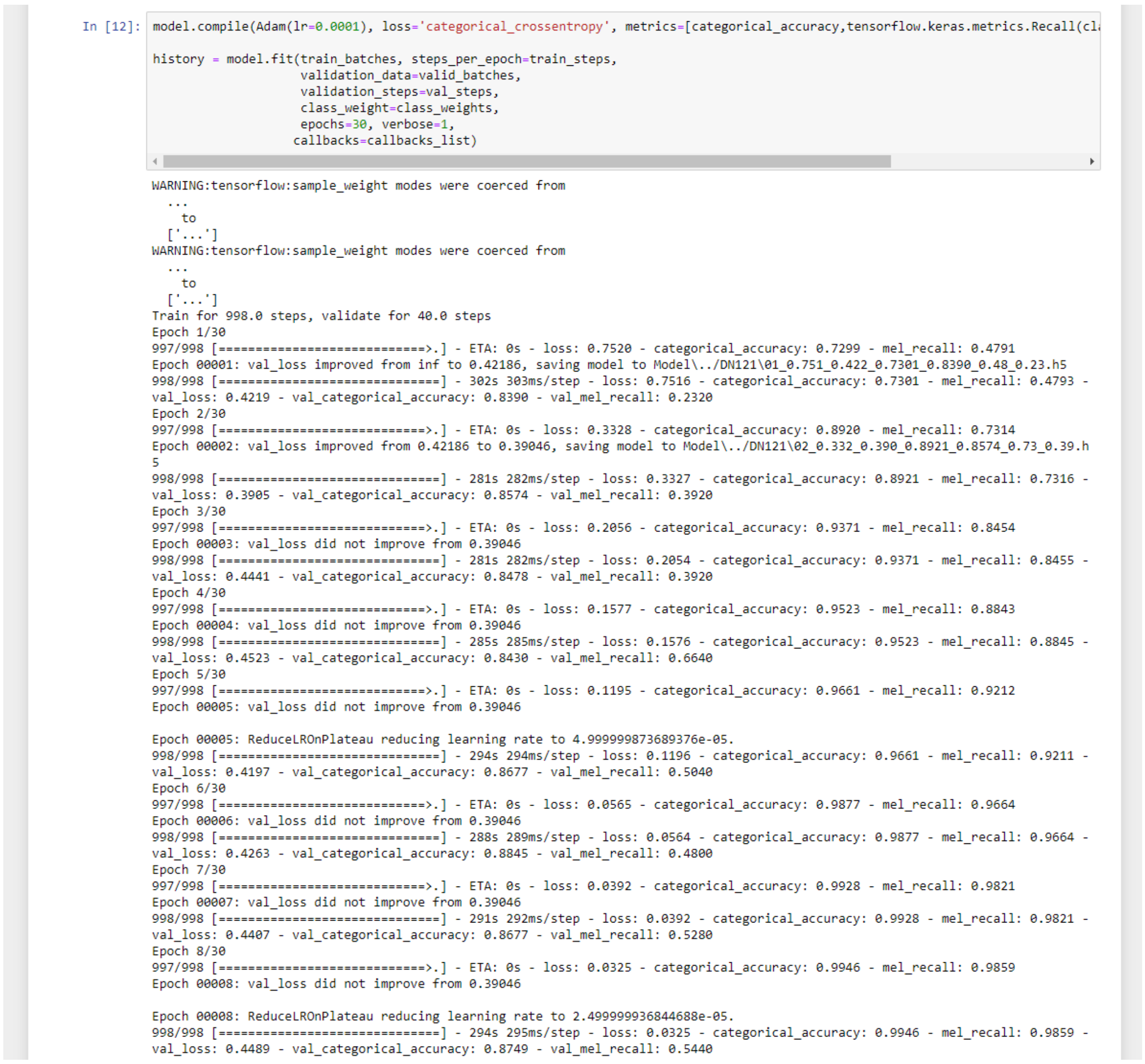
```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, None, None,  0

__________________________________________________________________________________________________
zero_padding2d (ZeroPadding2D)  (None, None, None, 3 0           input_1[0][0]
__________________________________________________________________________________________________
conv1/conv (Conv2D)             (None, None, None, 6 9408        zero_padding2d[0][0]
__________________________________________________________________________________________________
conv1/bn (BatchNormalization)   (None, None, None, 6 256         conv1/conv[0][0]
__________________________________________________________________________________________________
conv1/relu (Activation)         (None, None, None, 6 0           conv1/bn[0][0]
__________________________________________________________________________________________________
zero_padding2d_1 (ZeroPadding2D (None, None, None, 6 0           conv1/relu[0][0]
__________________________________________________________________________________________________
pool1 (MaxPooling2D)            (None, None, None, 6 0           zero_padding2d_1[0][0]
__________________________________________________________________________________________________
conv2_block1_0_bn (BatchNormali (None, None, None, 6 256         pool1[0][0]
```

Figure 40 CNN Fully Connected Layer Structure



```
In [9]: nv = os.path.join(train_path, 'nv')
        nv_folder = os.listdir(nv)
        nv_size = len(nv_folder)
        mel = os.path.join(train_path, 'mel')
        mel_folder = os.listdir(mel)
        mel_size = len(mel_folder)
        bkl = os.path.join(train_path, 'bkl')
        bkl_folder = os.listdir(bkl)
        bkl_size = len(bkl_folder)
        bcc = os.path.join(train_path, 'bcc')
        bcc_folder = os.listdir(bcc)
        bcc_size = len(bcc_folder)
        df = os.path.join(train_path, 'df')
        df_folder = os.listdir(df)
        df_size = len(df_folder)
        vasc = os.path.join(train_path, 'vasc')
        vasc_folder = os.listdir(vasc)
        vasc_size = len(vasc_folder)
        akiec = os.path.join(train_path, 'akiec')
        akiec_folder = os.listdir(akiec)
        akiec_size = len(akiec_folder)

        total = nv_size + mel_size + bkl_size + bcc_size + df_size + vasc_size + akiec_size
        class_weights={
            0: total/akiec_size/7, # akiec
            1: total/bcc_size/7, # bcc
            2: total/bkl_size/7, # bkl
            3: total/df_size/7, # df
            4: total/mel_size/7, # mel
            5: total/nv_size/7, # nv
            6: total/vasc_size/7, # vasc
        }

In [10]: model_folder = '../DN121'
         model_path = os.path.join('Model',model_folder)
         os.mkdir(model_path)
         filepath = os.path.join(model_path,"{epoch:02d}_{loss:.3f}_{val_loss:.3f}_{categorical_accuracy:02.4f}_{val_categorical_accuracy

In [11]: checkpoint = ModelCheckpoint(filepath, monitor='val_loss', verbose=1,
                                      save_best_only=True, mode='min')

         reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3,
                                        verbose=1, mode='auto', min_lr=0.00001)

         earlystop = EarlyStopping(monitor='val_loss', patience=9)


         callbacks_list = [checkpoint, reduce_lr, earlystop]
```

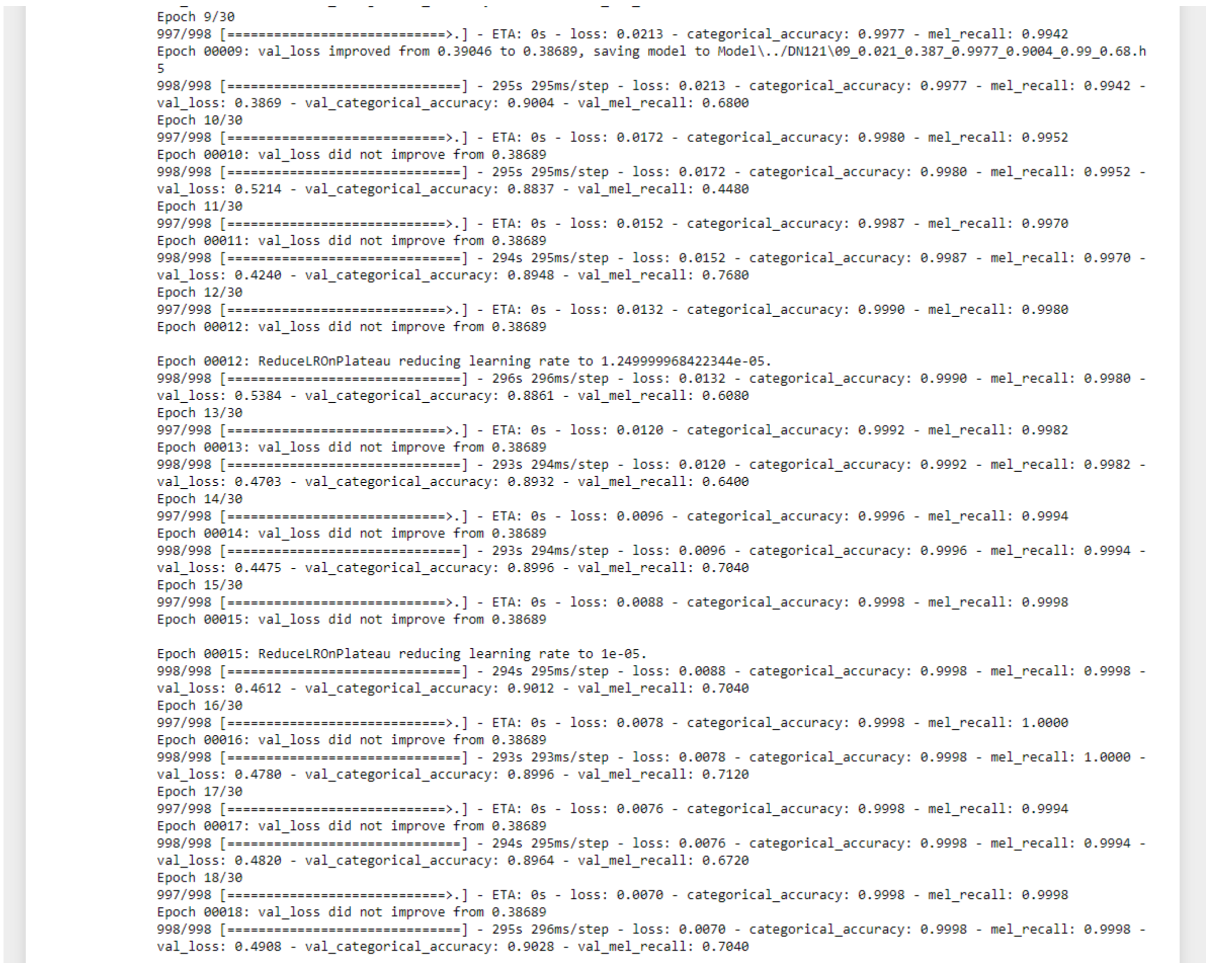
Figure 41 Training Setup



```
In [12]: model.compile(Adam(lr=0.0001), loss='categorical_crossentropy', metrics=[categorical_accuracy,tensorflow.keras.metrics.Recall(cl

         history = model.fit(train_batches, steps_per_epoch=train_steps,
                             validation_data=valid_batches,
                             validation_steps=val_steps,
                             class_weight=class_weights,
                             epochs=30, verbose=1,
                            callbacks=callbacks_list)
```

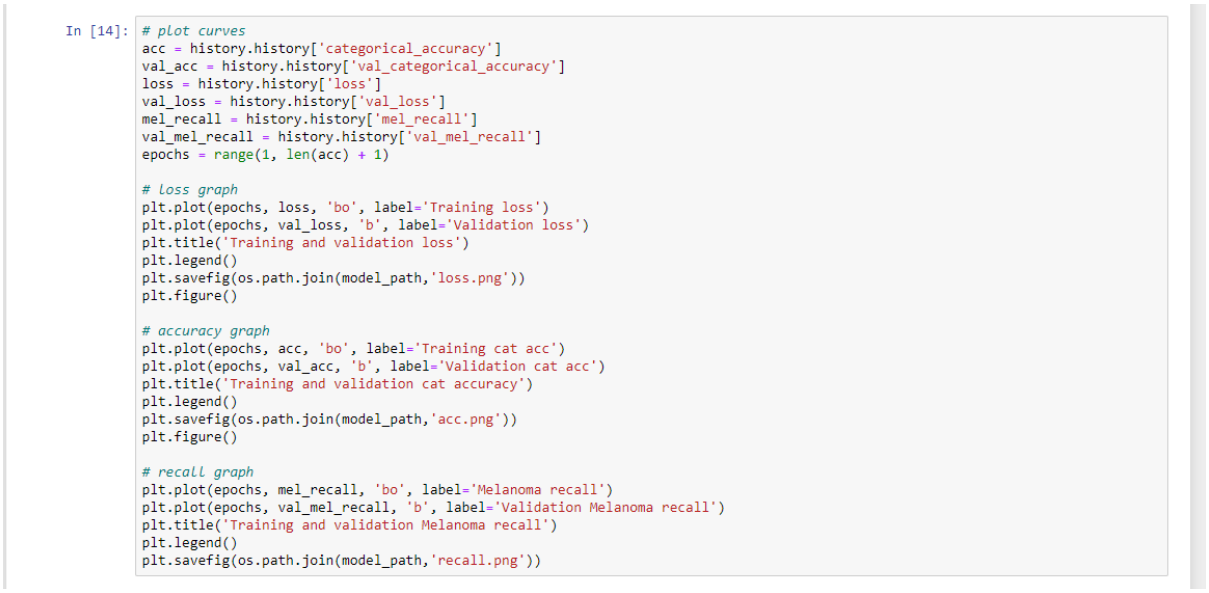
```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 998.0 steps, validate for 40.0 steps
Epoch 1/30
997/998 [============================>.] - ETA: 0s - loss: 0.7520 - categorical_accuracy: 0.7299 - mel_recall: 0.4791
Epoch 00001: val_loss improved from inf to 0.42186, saving model to Model\../DN121\01_0.751_0.422_0.7301_0.8390_0.48_0.23.h5
998/998 [==============================] - 302s 303ms/step - loss: 0.7516 - categorical_accuracy: 0.7301 - mel_recall: 0.4793 -
val_loss: 0.4219 - val_categorical_accuracy: 0.8390 - val_mel_recall: 0.2320
Epoch 2/30
997/998 [============================>.] - ETA: 0s - loss: 0.3328 - categorical_accuracy: 0.8920 - mel_recall: 0.7314
Epoch 00002: val_loss improved from 0.42186 to 0.39046, saving model to Model\../DN121\02_0.332_0.390_0.8921_0.8574_0.73_0.39.h
5
998/998 [==============================] - 281s 282ms/step - loss: 0.3327 - categorical_accuracy: 0.8921 - mel_recall: 0.7316 -
val_loss: 0.3905 - val_categorical_accuracy: 0.8574 - val_mel_recall: 0.3920
Epoch 3/30
997/998 [============================>.] - ETA: 0s - loss: 0.2056 - categorical_accuracy: 0.9371 - mel_recall: 0.8454
Epoch 00003: val_loss did not improve from 0.39046
998/998 [==============================] - 281s 282ms/step - loss: 0.2054 - categorical_accuracy: 0.9371 - mel_recall: 0.8455 -
val_loss: 0.4441 - val_categorical_accuracy: 0.8478 - val_mel_recall: 0.3920
Epoch 4/30
997/998 [============================>.] - ETA: 0s - loss: 0.1577 - categorical_accuracy: 0.9523 - mel_recall: 0.8843
Epoch 00004: val_loss did not improve from 0.39046
998/998 [==============================] - 285s 285ms/step - loss: 0.1576 - categorical_accuracy: 0.9523 - mel_recall: 0.8845 -
val_loss: 0.4523 - val_categorical_accuracy: 0.8430 - val_mel_recall: 0.6640
Epoch 5/30
997/998 [============================>.] - ETA: 0s - loss: 0.1195 - categorical_accuracy: 0.9661 - mel_recall: 0.9212
Epoch 00005: val_loss did not improve from 0.39046

Epoch 00005: ReduceLROnPlateau reducing learning rate to 4.999999873689376e-05.
998/998 [==============================] - 294s 294ms/step - loss: 0.1196 - categorical_accuracy: 0.9661 - mel_recall: 0.9211 -
val_loss: 0.4197 - val_categorical_accuracy: 0.8677 - val_mel_recall: 0.5040
Epoch 6/30
997/998 [============================>.] - ETA: 0s - loss: 0.0565 - categorical_accuracy: 0.9877 - mel_recall: 0.9664
Epoch 00006: val_loss did not improve from 0.39046
998/998 [==============================] - 288s 289ms/step - loss: 0.0564 - categorical_accuracy: 0.9877 - mel_recall: 0.9664 -
val_loss: 0.4263 - val_categorical_accuracy: 0.8845 - val_mel_recall: 0.4800
Epoch 7/30
997/998 [============================>.] - ETA: 0s - loss: 0.0392 - categorical_accuracy: 0.9928 - mel_recall: 0.9821
Epoch 00007: val_loss did not improve from 0.39046
998/998 [==============================] - 291s 292ms/step - loss: 0.0392 - categorical_accuracy: 0.9928 - mel_recall: 0.9821 -
val_loss: 0.4407 - val_categorical_accuracy: 0.8677 - val_mel_recall: 0.5280
Epoch 8/30
997/998 [============================>.] - ETA: 0s - loss: 0.0325 - categorical_accuracy: 0.9946 - mel_recall: 0.9859
Epoch 00008: val_loss did not improve from 0.39046

Epoch 00008: ReduceLROnPlateau reducing learning rate to 2.499999936844688e-05.
998/998 [==============================] - 294s 295ms/step - loss: 0.0325 - categorical_accuracy: 0.9946 - mel_recall: 0.9859 -
val_loss: 0.4489 - val_categorical_accuracy: 0.8749 - val_mel_recall: 0.5440
```

Figure 42 Training Process



```
Epoch 9/30
997/998 [============================>.] - ETA: 0s - loss: 0.0213 - categorical_accuracy: 0.9977 - mel_recall: 0.9942
Epoch 00009: val_loss improved from 0.39046 to 0.38689, saving model to Model\../DN121\09_0.021_0.387_0.9977_0.9004_0.99_0.68.h
5
998/998 [==============================] - 295s 295ms/step - loss: 0.0213 - categorical_accuracy: 0.9977 - mel_recall: 0.9942 -
val_loss: 0.3869 - val_categorical_accuracy: 0.9004 - val_mel_recall: 0.6800
Epoch 10/30
997/998 [============================>.] - ETA: 0s - loss: 0.0172 - categorical_accuracy: 0.9980 - mel_recall: 0.9952
Epoch 00010: val_loss did not improve from 0.38689
998/998 [==============================] - 295s 295ms/step - loss: 0.0172 - categorical_accuracy: 0.9980 - mel_recall: 0.9952 -
val_loss: 0.5214 - val_categorical_accuracy: 0.8837 - val_mel_recall: 0.4480
Epoch 11/30
997/998 [============================>.] - ETA: 0s - loss: 0.0152 - categorical_accuracy: 0.9987 - mel_recall: 0.9970
Epoch 00011: val_loss did not improve from 0.38689
998/998 [==============================] - 294s 295ms/step - loss: 0.0152 - categorical_accuracy: 0.9987 - mel_recall: 0.9970 -
val_loss: 0.4240 - val_categorical_accuracy: 0.8948 - val_mel_recall: 0.7680
Epoch 12/30
997/998 [============================>.] - ETA: 0s - loss: 0.0132 - categorical_accuracy: 0.9990 - mel_recall: 0.9980
Epoch 00012: val_loss did not improve from 0.38689

Epoch 00012: ReduceLROnPlateau reducing learning rate to 1.249999968422344e-05.
998/998 [==============================] - 296s 296ms/step - loss: 0.0132 - categorical_accuracy: 0.9990 - mel_recall: 0.9980 -
val_loss: 0.5384 - val_categorical_accuracy: 0.8861 - val_mel_recall: 0.6080
Epoch 13/30
997/998 [============================>.] - ETA: 0s - loss: 0.0120 - categorical_accuracy: 0.9992 - mel_recall: 0.9982
Epoch 00013: val_loss did not improve from 0.38689
998/998 [==============================] - 293s 294ms/step - loss: 0.0120 - categorical_accuracy: 0.9992 - mel_recall: 0.9982 -
val_loss: 0.4703 - val_categorical_accuracy: 0.8932 - val_mel_recall: 0.6400
Epoch 14/30
997/998 [============================>.] - ETA: 0s - loss: 0.0096 - categorical_accuracy: 0.9996 - mel_recall: 0.9994
Epoch 00014: val_loss did not improve from 0.38689
998/998 [==============================] - 293s 294ms/step - loss: 0.0096 - categorical_accuracy: 0.9996 - mel_recall: 0.9994 -
val_loss: 0.4475 - val_categorical_accuracy: 0.8996 - val_mel_recall: 0.7040
Epoch 15/30
997/998 [============================>.] - ETA: 0s - loss: 0.0088 - categorical_accuracy: 0.9998 - mel_recall: 0.9998
Epoch 00015: val_loss did not improve from 0.38689

Epoch 00015: ReduceLROnPlateau reducing learning rate to 1e-05.
998/998 [==============================] - 294s 295ms/step - loss: 0.0088 - categorical_accuracy: 0.9998 - mel_recall: 0.9998 -
val_loss: 0.4612 - val_categorical_accuracy: 0.9012 - val_mel_recall: 0.7040
Epoch 16/30
997/998 [============================>.] - ETA: 0s - loss: 0.0078 - categorical_accuracy: 0.9998 - mel_recall: 1.0000
Epoch 00016: val_loss did not improve from 0.38689
998/998 [==============================] - 293s 293ms/step - loss: 0.0078 - categorical_accuracy: 0.9998 - mel_recall: 1.0000 -
val_loss: 0.4780 - val_categorical_accuracy: 0.8996 - val_mel_recall: 0.7120
Epoch 17/30
997/998 [============================>.] - ETA: 0s - loss: 0.0076 - categorical_accuracy: 0.9998 - mel_recall: 0.9994
Epoch 00017: val_loss did not improve from 0.38689
998/998 [==============================] - 294s 295ms/step - loss: 0.0076 - categorical_accuracy: 0.9998 - mel_recall: 0.9994 -
val_loss: 0.4820 - val_categorical_accuracy: 0.8964 - val_mel_recall: 0.6720
Epoch 18/30
997/998 [============================>.] - ETA: 0s - loss: 0.0070 - categorical_accuracy: 0.9998 - mel_recall: 0.9998
Epoch 00018: val_loss did not improve from 0.38689
998/998 [==============================] - 295s 296ms/step - loss: 0.0070 - categorical_accuracy: 0.9998 - mel_recall: 0.9998 -
val_loss: 0.4908 - val_categorical_accuracy: 0.9028 - val_mel_recall: 0.7040
```

Figure 43 Training Process (Continue)



```
In [14]: # plot curves
         acc = history.history['categorical_accuracy']
         val_acc = history.history['val_categorical_accuracy']
         loss = history.history['loss']
         val_loss = history.history['val_loss']
         mel_recall = history.history['mel_recall']
         val_mel_recall = history.history['val_mel_recall']
         epochs = range(1, len(acc) + 1)

         # loss graph
         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.savefig(os.path.join(model_path,'loss.png'))
         plt.figure()

         # accuracy graph
         plt.plot(epochs, acc, 'bo', label='Training cat acc')
         plt.plot(epochs, val_acc, 'b', label='Validation cat acc')
         plt.title('Training and validation cat accuracy')
         plt.legend()
         plt.savefig(os.path.join(model_path,'acc.png'))
         plt.figure()

         # recall graph
         plt.plot(epochs, mel_recall, 'bo', label='Melanoma recall')
         plt.plot(epochs, val_mel_recall, 'b', label='Validation Melanoma recall')
         plt.title('Training and validation Melanoma recall')
         plt.legend()
         plt.savefig(os.path.join(model_path,'recall.png'))
```

Figure 44 Plotting Graphs

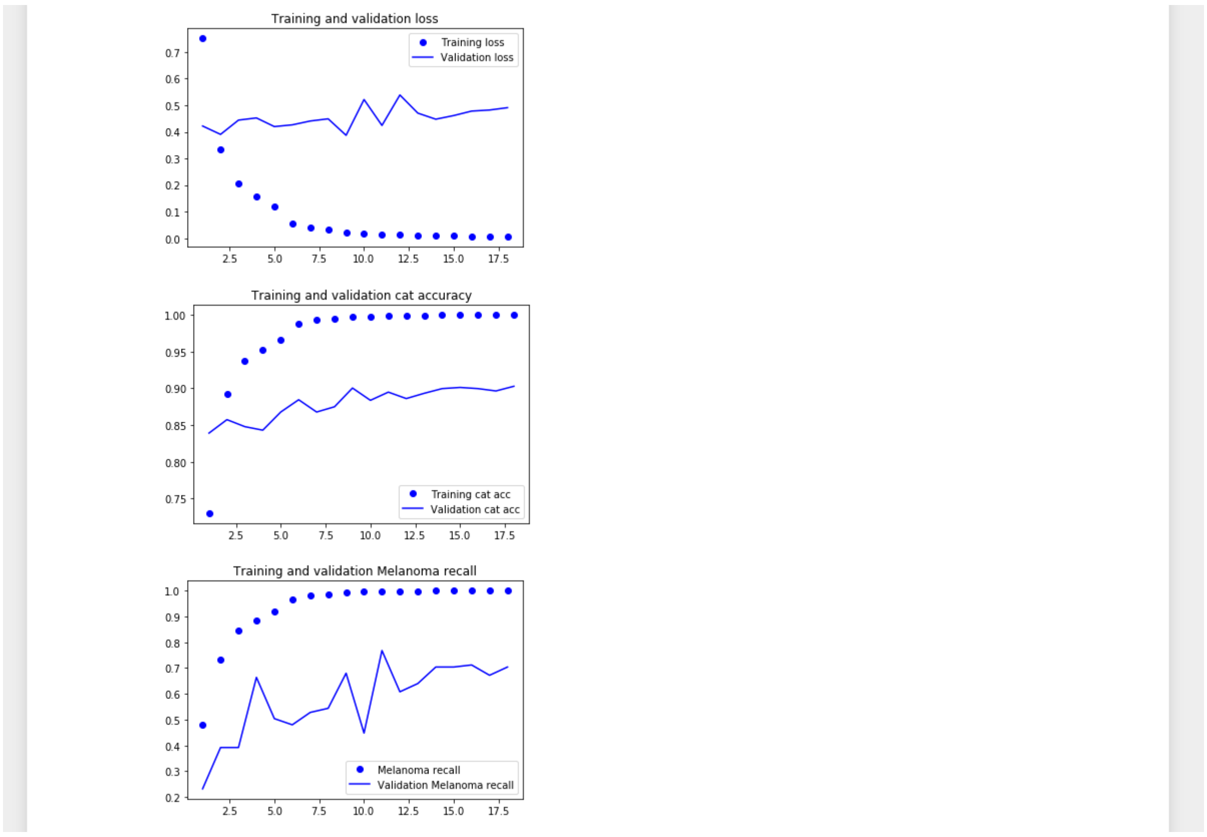
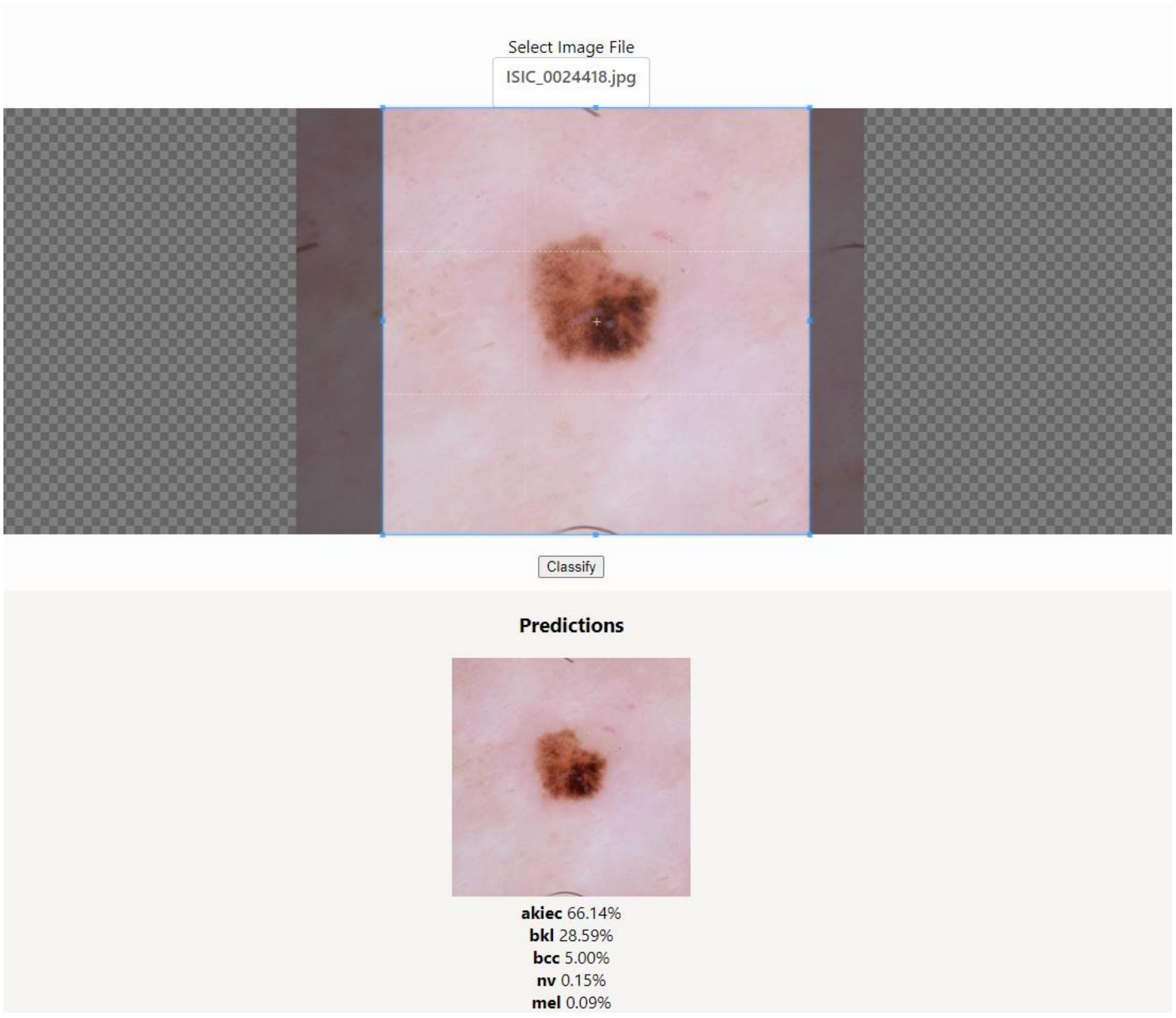


Figure 45 Graphs Plotted



Predictions

akiec 66.14%
bkl 28.59%
bcc 5.00%
nv 0.15%
mel 0.09%

Figure 46 Website UI & Example of Classification Result

References

[1] H. Younis, M. H. Bhatti, and M. Azeem, "Classification of skin cancer dermoscopy images using transfer learning," *15th Int. Conf. Emerg. Technol. ICET 2019*, 2019, doi: 10.1109/ICET48972.2019.8994508.

[2] A. Aggarwal, N. Das, and I. Sreedevi, "Attention-guided deep convolutional neural networks for skin cancer classification," *2019 9th Int. Conf. Image Process. Theory, Tools Appl. IPTA 2019*, no. July 2020, 2019, doi: 10.1109/IPTA.2019.8936100.

[3] P. Tschandl, C. Rosendahl, and H. Kittler, "Data descriptor: The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Sci. Data*, vol. 5, pp. 1–9, 2018, doi: 10.1038/sdata.2018.161.

[4] A. Esteva and E. Topol, "Can skin cancer diagnosis be transformed by AI?," *Lancet*, vol. 394, no. 10211, p. 1795, 2019, doi: 10.1016/S0140-6736(19)32726-6.

[5] A. Chahal and P. Gulia, "Machine learning and deep learning," *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 12, pp. 4910–4914, 2019, doi: 10.35940/ijitee.L3550.1081219.

[6] S. Panigrahi, A. Nanda, and T. Swarnkar, "A Survey on Transfer Learning," *Smart Innov. Syst. Technol.*, vol. 194, no. 10, pp. 781–789, 2021, doi: 10.1007/978-981-15-5971-6_83.

[7] L. Shao, F. Zhu, and X. Li, "Transfer learning for visual categorization: A survey," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 26, no. 5, pp. 1019–1034, 2015, doi: 10.1109/TNNLS.2014.2330900.

[8] N. Sharma, V. Jain, and A. Mishra, "An Analysis of Convolutional Neural Networks for Image Classification," *Procedia Comput. Sci.*, vol. 132, no. Iccids, pp. 377–384, 2018, doi: 10.1016/j.procs.2018.05.198.

[9] G. Kim and C. Lee, "Convolutional Neural Network 를 이용한 한국어 영화평 감성 분석 Using Convolutional Neural Network," vol. 2644, no. January, pp. 747–749, 2016, doi: 10.13140/RG.2.2.29424.69127.

[10] G. Habib and S. Qureshi, "Optimization and acceleration of convolutional neural networks: A survey," *J. King Saud Univ. - Comput. Inf. Sci.*, 2020, doi: 10.1016/j.jksuci.2020.10.004.

[11] R. Xi, M. Hou, M. Fu, H. Qu, and D. Liu, "Deep Dilated Convolution on Multimodality Time Series for Human Activity Recognition," *Proc. Int. Jt. Conf. Neural Networks*, vol. 2018-July, 2018, doi: 10.1109/IJCNN.2018.8489540.

[12] "Actinic keratosis — Know It All!. All you need to know about Actinic… | by Clipo | Medium." https://clipo-in.medium.com/actinic-keratosis-know-it-all-d85b0e74ddfb (accessed Apr. 25, 2021).

[13] "Basal Cell Cancer Symptoms. Basal cell cancer, the most common type… | by David Eltz | Medium." https://medium.com/@eltz.david/basal-cell-cancer-symptoms-ed15ee8e87db (accessed Apr. 25, 2021).

[14] "Seborrheic keratosis - Symptoms and causes - Mayo Clinic." https://www.mayoclinic.org/diseases-conditions/seborrheic-keratosis/symptoms-causes/syc-20353878 (accessed Apr. 25, 2021).

[15] "Dermatofibroma Guide: Causes, Symptoms and Treatment Options." https://www.drugs.com/health-guide/dermatofibroma.html (accessed Apr. 25, 2021).

[16] "Melanoma - Symptoms and causes - Mayo Clinic." https://www.mayoclinic.org/diseases-

conditions/melanoma/symptoms-causes/syc-20374884 (accessed Apr. 25, 2021).

[17] “Melanocytic Nevus | Skin Lesion | Know Cancer.” https://www.knowcancer.com/oncology/melanocytic-nevus/ (accessed Apr. 25, 2021).

[18] “Pyogenic Granuloma: Causes, Diagnosis, and Treatments.” https://www.healthline.com/health/pyogenic-granuloma#complications (accessed Apr. 25, 2021).

[19] M. Abadi *et al.*, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems,” 2016, [Online]. Available: http://arxiv.org/abs/1603.04467.

[20] D. Rolon-Mérette, M. Ross, T. Rolon-Mérette, and K. Church, “Introduction to Anaconda and Python: Installation and setup,” *Quant. Methods Psychol.*, vol. 16, no. 5, pp. S3–S11, 2020, doi: 10.20982/tqmp.16.5.s003.

[21] C. Li and C. Li, “Web Front-End Realtime Face Recognition Based on TFJS,” *Proc. - 2019 12th Int. Congr. Image Signal Process. Biomed. Eng. Informatics, CISP-BMEI 2019*, 2019, doi: 10.1109/CISP-BMEI48845.2019.8965963.

[22] D. Smilkov *et al.*, “Tensorflow.JS: Machine learning for the web and beyond,” *arXiv*, 2019.

[23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 2818–2826, 2016, doi: 10.1109/CVPR.2016.308.

[24] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[25] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 2261–2269, 2017, doi: 10.1109/CVPR.2017.243.

[26] Z. Wan, Z. Yuxiang, X. Gong, Zhanghuali, and B. Yu, “DenseNet model with RAdam optimization algorithm for cancer image classification,” *2021 IEEE Int. Conf. Consum. Electron. Comput. Eng. ICCECE 2021*, no. Iccece, pp. 771–775, 2021, doi: 10.1109/ICCECE51280.2021.9342268.

[27] “Keras Applications.” https://keras.io/api/applications/ (accessed Apr. 25, 2021).

[28] G. J. Chowdary, N. S. Punn, S. K. Sonbhadra, and S. Agarwal, “Face mask detection using transfer learning of inceptionV3,” *arXiv*, pp. 1–10, 2020, doi: 10.1007/978-3-030-66665-1_6.

[29] T. Y. Hsiao, Y. C. Chang, H. H. Chou, and C. Te Chiu, “Filter-based deep-compression with global average pooling for convolutional networks,” *J. Syst. Archit.*, vol. 95, pp. 9–18, 2019, doi: 10.1016/j.sysarc.2019.02.008.

[30] “Keras Normalization Layers- Batch Normalization and Layer Normalization Explained for Beginners | MLK - Machine Learning Knowledge.” https://machinelearningknowledge.ai/keras-normalization-layers-explained-for-beginners-batch-normalization-vs-layer-normalization/ (accessed Apr. 25, 2021).

[31] H. O. A. Ssignment, “H a #2,” *Response*, pp. 3–5, 2011.

[32] A. M. Javid, S. Das, M. Skoglund, and S. Chatterjee, “A ReLU Dense Layer to Improve the Performance of Neural Networks,” no. October, 2020, [Online]. Available: http://arxiv.org/abs/2010.13572.

[33] “Softmax Activation Function with Python.” https://machinelearningmastery.com/softmax-activation-function-with-python/ (accessed Apr. 25, 2021).

[34] M. Rahimzadeh and A. Attar, “a New Modified Deep Convolutional Neural Network for Detecting Covid-19 From X-Ray Images,” *arXiv*, vol. 19, no. May, p. 100360, 2020, doi: 10.1016/j.imu.2020.100360.

[35] J. M. Ashfaque, “Confusion Matrix Demystified Johar M . Ashfaque,” no. August, 2020.