A Model for Classification of Seven types of Skin Cancer Stage using a Novel CNN Technique

Submitted to Indian Institute of Technology (ISM), Dhanbad in fulfillment of the requirements for final year project

of

B.Tech

in

Computer Science & Engineering

by

Vallela Kaushik Shashank Reddy

Admission No.:16JE002661

Under the guidance of Dr. Annavarapu Chandra Sekhara Rao

Assistant Professor

(Dept. of Computer Science and Engineering)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Indian Institute of Technology (Indian School of Mines), Dhanbad-826004

APRIL 2020

**CERTIFICATE**

This is to certify that the project entitled “**A Model for Classification of Seven types of Skin Cancer Stage using Novel CNN Technique**” is a record of the investigation carried out by Vallela Kaushik Shashank Reddy, Admission No. 16JE002661, under the guidance of Dr.Annavarapu Chandra Sekhara Rao.

This report is submitted in fulfillment for the requirement of Final Year Project of the Winter Semester during the academic session 2019 to 2020.Indian Institute of Technology (Indian School of Mines), Dhanbad.

The result embodied in this report has not been submitted for the award of any other purposes or requirements.

…………………………… .…………………………

Dr. Haider Banka Dr. Annavarapu Chandra Sekhara Rao

(Associate Professor) (Assistant Professor) Head of Department Department of CSE Department of CSE IIT (ISM) Dhanbad-826004 IIT (ISM) Dhanbad-826004

**DECLARATION**

I hereby declare that this thesis is an authenticate record of the research work carried out by

me. To the best of my knowledge it contains no material previously published or written by

another person or material which has been accepted for the award of any degree or diploma

of the university or other institutes of higher learning, where due acknowledgement has been

made in the text.

Date: April,2020 Vallela Kaushik Shashank Reddy

Admission No: 16JE002661

B.Tech (Computer Science & Engineering)

Dept. of Computer Science & Engineering

Indian Institute of Technology(ISM), Dhanbad

**ACKNOWLEDGEMENT**

I would like to express my deep sense of gratitude towards Dr. Annavarapu Chandra Sekhara Rao Associate Professor, Department of Computer Science and Engineering, Indian Institute of Technology (Indian School of Mines), Dhanbad, under his guidance I have learnt many things. He has always been inspiring and has always catered to my doubts and problems with solutions. I thank you sir for your guidance.

I would also express my gratitude towards Samson Anosh sir, Research Scholar, Department of Computer Science and Engineering, Indian Institute of Technology (Indian School of Mines), Dhanbad, for his support and for giving his time.

I express my gratitude towards all the faculty members for their encouragement, support and blessings.

I would also like to express my gratitude towards my batch mates for their support and well wishes.

Lastly, I would like to thank my family members for their constant support and inspiration and for having enormous faith in me.

Vallela Kaushik Shashank Reddy Admission No: 16JE002661 B.Tech(Computer Science and Engineering)

IIT(ISM) Dhanbad.

**Abstract**

Skin Cancer is the most common type of malignancy seen in humans, it is primarily diagnosed by visually looking at affected areas. Diagnosis begins with initial screening and then by dermoscopic analysis, a biopsy and then by histopathological examination. Automatic classification using machine learning and deep learning techniques is a challenging task due to the fine grained difference in the appearance of skin lesions.( A **skin lesion** is a part of the **skin** that has an abnormal growth or appearance compared to the **skin** around it).There are different types of skin cancer and treatment for each type is different. So identification of type of skin cancer is important. Automated process helps to make diagnosis easy.

In this paper 7 different types of skin cancer are predicted. They are:

1. **Melanocytic nevi**
2. **Melanoma**
3. **Benign keratosis-like lesions**
4. **Basal cell carcinoma**
5. **Actinic keratoses**
6. **Vacula lesions**
7. **Dermatofibroma**

The dataset used for the project is collected from kaggle. This dataset contains more than 10000 images of 7 different types of cancer.

dataset link: <https://www.kaggle.com/kmader/skin-cancer-mnist-ham10000>

Original Data sources as mentioned in kaggle are

<https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T>

[1] Noel Codella, Veronica Rotemberg, Philipp Tschandl, M. Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, Harald Kittler, Allan Halpern: “Skin Lesion Analysis Toward Melanoma Detection 2018: A Challenge Hosted by the International Skin Imaging Collaboration (ISIC)”, 2018;

<https://arxiv.org/abs/1902.03368>

[2] Tschandl, P., Rosendahl, C. & Kittler, H. The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. Sci. Data 5, 180161 doi:10.1038/sdata.2018.161 (2018).

Contents

Certificate ……………………………………………………………….……....i

Acknowledgements …………………………………………………………….ii

Abstract ………………………………………………………………………...iii

1 Introduction …………………………………………………….…………………………………………..1

1.1 Introduction

1.2 Related Discussion

2 Exploratory Data Analysis(EDA) ..………………….…………………………………………..2

2.1 Feature wise study wrt count

2.2 Comparison with lesion type

3 Data Processing …………………………………………………………………………………...........3

3.1 Data Cleaning

3.2 Resizing Images

3.3 Data Normalization

3.4 Data Balancing

3.5 Data Augmentation

4 Image Classification Using CNN ………………………………………………………...........4

4.1 Why CNN for skin cancer image classification?

4.2 Different Concepts used

4.3 Model Building

4.4 Training the Model

5 Results ….….………………………………………………………………….5

7 References ……………………………………………………………...….….7

**Chapter 1**

**Introduction**

**1.1 Introduction**

Deep learning is a branch of machine learning that models high-level abstractions in data using many processing layers. In this paper one such model CNN is discussed and used. It was initially designed to recognise cursive numbers and is later proved to be useful in object detection. These models are proved to be powerful classification tools. CNN models such as VGG, GoogleNet and ResNethave reported very good performances in image classification and recognition.

Despite the advancements in technology, however the inefficiency in the clinical dataset has limited the application of deep learning in biological data.

Melanoma is a common skin cancer that has high mortality rate. It is estimated that nearly 9,730 deaths have occured due to melanoma in 2017.

Basal cell carcinoma commonly known as BCC is the most common skin cancer but is usually not fatal.

So it is very important for both health care services to diagnose the type of cancer and to develop an efficient method to discriminate different types of skin cancer would therefore be benificial for initial screening. In this study, I have used a novel CNN algorithm (Snapshot Ensemble with Resnet50) in an attempt to develop an automated classification tool using biological images of 7 different established skin lesions - Melanocytic nevi, Melanoma, Benign keratosis, Basal cell carcinoma, Actinic keratoses, Vascular lesions, Dermatofibroma.

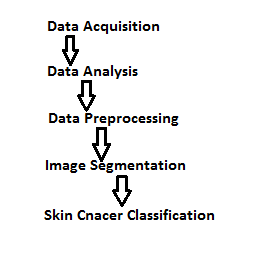


Fig.1: Flow used in Skin Cancer Diagnosis

In this study we used CNN based novel architecture using Snapshot Ensemble upon resnet50 architecture to classify the skin lesions. with the application of normal CNN architecture I got an accuracy of 75.43% (+/- 1%) with the validation set and an accuracy of 76.14%(+/- 1%) for test set. When **Snapshot Ensemble**(will be discussed below) was used the final accuracy was risen to **93.01%** for test set. This improvement is achived by implementing all the functions in the flow diagram given in [ Fig.1].

**1.2 Related Discussion**

Considerable efforts are to be made to develop an automated image classification systems for the precise detection of lesions. In an old study, Computer-aided diagnostic methods depending on a feature extaction algorithm imaged a considerable diagnostic ability with certain types of skin cancer, which includes melanoma. However, a human-engineered AI algorithm could not make accurate diagnoses over a broader class of skin diseases. In recent years , deep CNN architectures became popular in feature learning and object classification especially with image data. Extensive reasearch from ImageNet Large Scale Visual Recognition Challenge has indicated that object classification abilities of CNNs can exceed over those of human diagnosis abilities.

Several dermatologic studies reported the uses of machine learning or deep learning. For example, Liao et ai. trained a CNN to classify 23 top-level categories such as bullous disease, viral infections, etc. using 23000 images. It showed an accuracy of 73.1% and 91.0% respectively for rates at which a model outputs the correct label with its top-1 and top-5 predictions for a given image. The performance of the binary (benign/malignant) classification method used by the CNN system in that report was on par with that of all of the dermatologists who participated. The authors determined an AUC of 0.96 for the diagnosis of carcinoma in 707 cases from the Edinburgh dataset and of 0.96 for the diagnosis of melanoma using 225 cases.

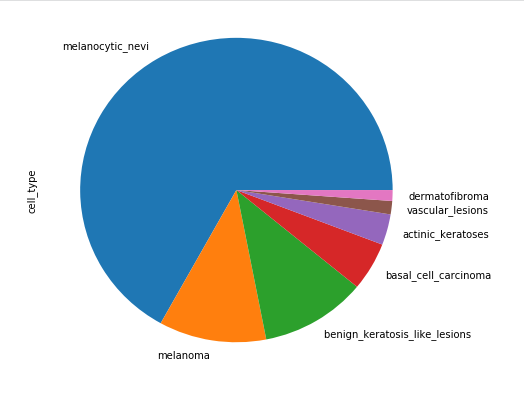
**Chapter 2**

**Exploratory Data Analysis(EDA)**

Here we discuss about the different features o the dataset, their distributions and the count of that types present in the dataset. This is helpful to analyse the nature of our data and helps us in the data processing step. First we will see the number of instances of data present for every possible values of every feature of data i.e. feature wise study of the data.

**2.1 Feature wise study wrt count**

**1) cell\_type**

**Fig 2.1**

This is a pie chart based on number of instances of particular lesion type present in data . This clearly shows that in the data set cell type Melanocytic nevi has very large number of instances in comparison to other cell types.

**2) Technical Validation field (ground truth) which is dx\_type**

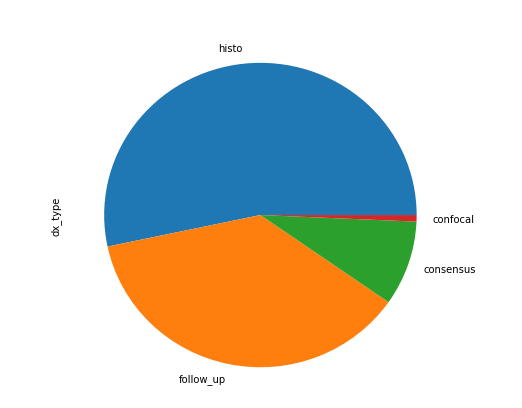
This field contains four catogeries,

**1. Histopathology(Histo):** Histopathologic diagnoses of excised lesions have been performed by specialized dermatopathologists.

**2. Confocal:** Reflectance confocal microscopy is an in-vivo imaging technique with a resolution at near-cellular level , and some facial benign with a grey-world assumption of all training-set images in Lab-color space before and after manual histogram changes.

**3. Follow-up:** If nevi monitored by digital dermatoscopy did not show any changes during 3 follow-up visits or 1.5 years biologists accepted this as evidence of biologic benignity. Only nevi, but no other benign diagnoses were labeled with this type of ground-truth because dermatologists usually do not monitor dermatofibromas, seborrheic keratoses, or vascular lesions.

**4. Consensus:** For typical benign cases without histopathology or followup biologists provide an expert-consensus rating of authors PT and HK. They applied the consensus label only if both authors independently gave the same unequivocal benign diagnosis. Lesions with this type of groundtruth were usually photographed for educational reasons and did not need further follow-up or biopsy for confirmation.

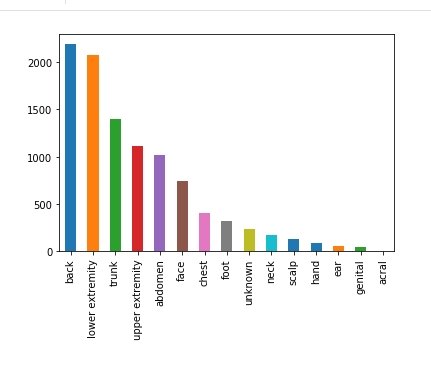
****

**Fig 2.2**

This pie chart shows that the data set contains more than half portion of data from Histopathology.

**3) Localization**

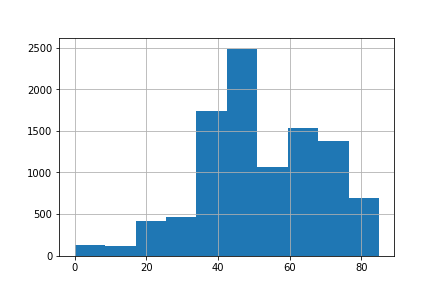
This field describes about the physical area from which the samples are collected from individuals.



**Fig 2.3**

This bar graph shows back , lower extremity, trunk and upper extremity are four heavily affected regions of skin cancer.

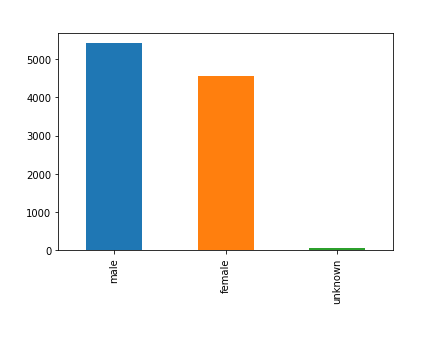
**4) Age**

****

**Fig 2.4**

It is clear that most effected people by skin cancer are between 35 and 50 years of age.

**5) Sex**

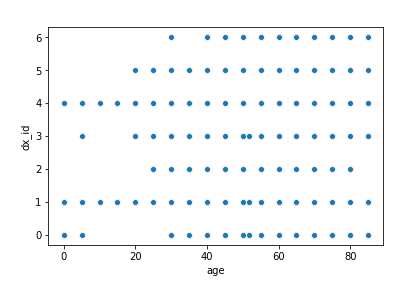
****

**Fig 2.5**

Not much to differentiate here with this bar graph.

**2.2 Comparison with lesion type**

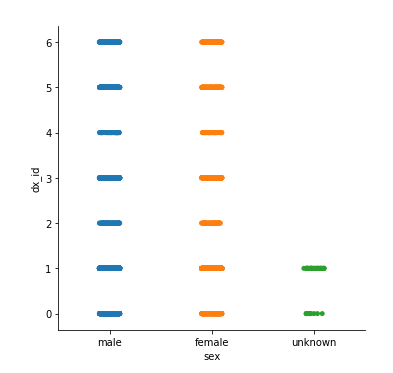
**1)**  **Age wise distribution of skin cancer types**

****

**Fig 2.6**

This scatter plot reveals that skin cancer types 2, 3, 5 and 6 which are Melanocytic nevi, Dermatofibroma, Basal cell carcinoma and Vascular lesions are not seen much below the age of 20 years.

**2) Sex wise distribution of skin cancer type**

****

**Fig 2.7**

This cat-plot shows that there are no skin cancer types which are specific to a particular sex, that any of 7 skin cancer types can be seen in both males and females.

This concludes the data analysis part. we have analyzed the data clearly and now with this knowledge, data cleaning and preprocessing should be applied efficiently for a optimal training of dataset with our model.

**Chapter 3**

**Data Processing**

The next stage before training the data is data processing. Various steps applied here are,

1) Data Cleaning

2) Resizing Images

3) Data Normalization

4) Data Balancing

5) Data Augmentation

Each of these steps are discussed below.

**3.1 Data Cleaning**

Data cleaning is a important step in machine learning. It plays an important role in building a model. Proper data cleaning can make or break the project. There is a popular belief that "Better data beats fancier algorithms".

Different steps involved in data cleaning are,

**1) Removal of unwanted observations**

This includes deleting duplicate or redundant or irrelevant values from dataset. Duplicate data arise most frequently during data collection and Irrelevant observations are those that don’t actually fit the specific problem that you’re trying to solve**.** Our data is does not contain any unwanted or duplicate, so we ignore this step.

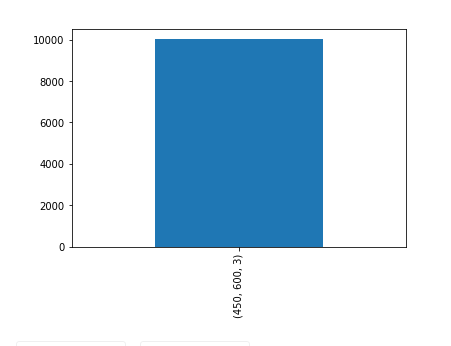
**2) Handling missing data**

This type of data is a tricky issue in machine learning. This data cannot be ignored or removed from dataset because this data may contain important features specific to the corresponding class other than that of missing feature.

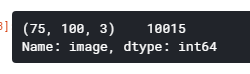
In this step we find all the null values in the data and replace them with mean values of that field. In our data only '**age**' field has null data as shown in and they are replaced by the mean values.

**3.2 Resizing Images**

original images in the dataset are all of same size but the size is very large. The images are of dimension (450 x 600 x 3) as seen in [Fig 3.1]. This will have a huge computation time while training and Tensorflow cannot handle this.

**Fig 3.1**

Let us resize the images by keeping the same size ratio so that no information is lost. I have resized the images to dimension (75 x 100 x 3). Now the shape of all images and their count is shown in [Fig 3.2].



**Fig 3.2**

**3.3 Data Normalization**

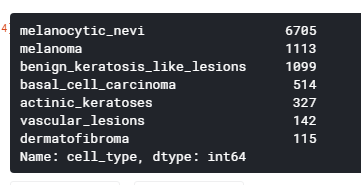
Original images are represented in colour code format with 3 values of Red, Blue and Green for each pixel and each value ranging from 0 to 255. So the normalization is applied as follows,

Normalized Image = ( Original Image - mean( all original images)) / standard\_deviation( all original images)

After normalization, each value of colour code format changed to a range of -2 to 2 which are preferred by neural networks.

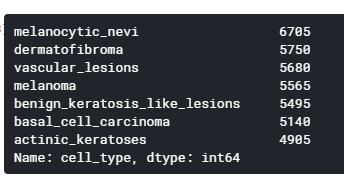
**3.4 Data Balancing**

we have seen from data analysis that the data is highly **unbalanced**. Number of instances of each lesion type present in original data are shown in [Fig 3.3].



**Fig 3.3: Unbalanced Data Category Count**

This would make the model more or less biased towards certain classes. To solve this issue, I artificially added images to fewer categories to make them as equal as those of largest class. To **resample** these, we randomly chose and copy the images of fewer class. This will create duplicate images. To address this issue we have to perform **data augmentation** which will be discussed below. After balancing the number of instances of each categories are shown in [Fig 3.4].



**Fig 3.4: Balanced Data Category Count**

**3.5 Data Augmentation**

Deep neural networks perform better with large amount of data. Aim of this step is to create images that depict the features of its class in every possible angle. This makes sure that at whatever angle the image may be taken, our trained model can predict it with more precision.

Different techniques used for this are,

1) Randomly rotate images in the range (degrees, 0 to 180)

2) Randomly zoom image

3) Randomly shift images horizontally

4) Randomly shift images vertically

5) Randomly flip the images horizontally

6) Randomly flip the images vertically

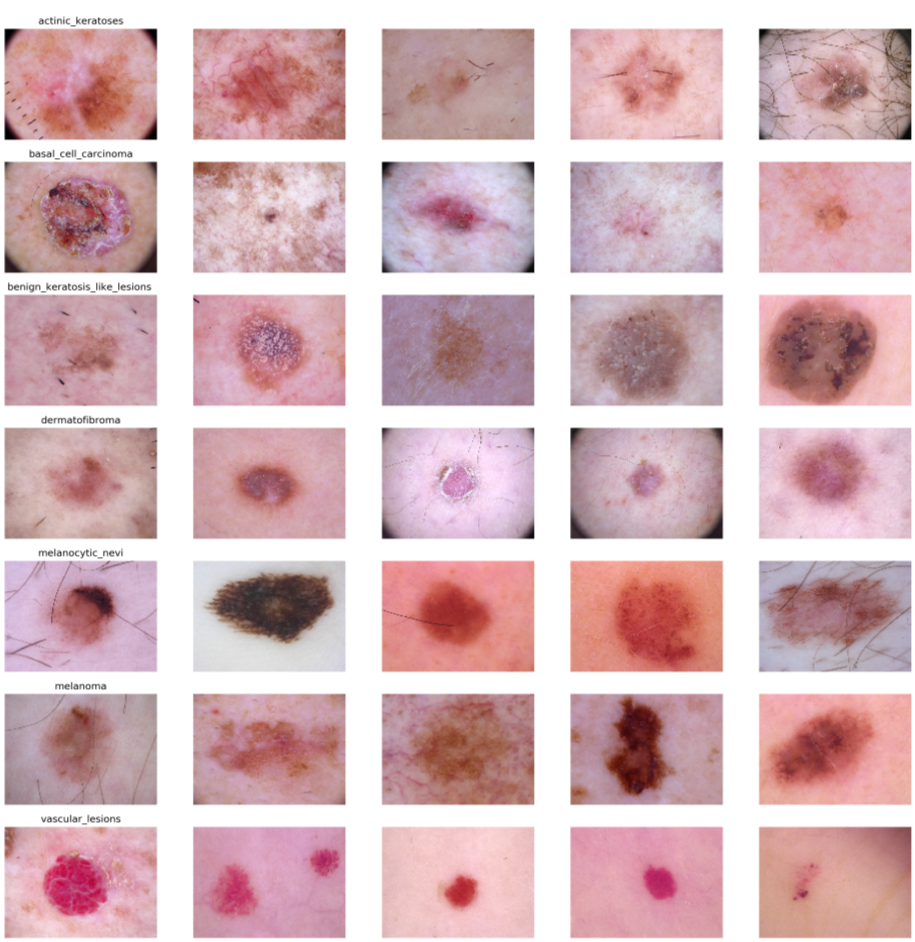
Before starting the training, the dataset is divided into training data(80%) and test data(20%). And this training data is again divided into training data(80%) and validation data(20%). The data augmentation is only performed on the finally obtained training data, Because only this data is used for training the model.

**Chapter 4**

**Image Classification using CNN**

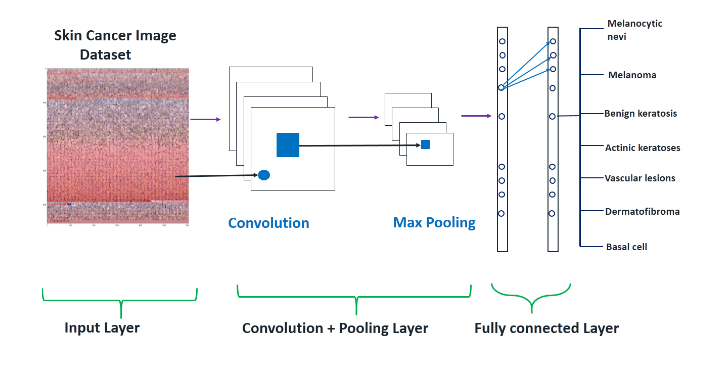
**4.1 Why CNN for skin cancer image classification?**

Convolution Neural Networks are widely used for image classification. They tend to produce better results than other classification algorithms because of their efficiency to handle large data. CNN uses a unique way of image classification by applying different filters to find the specific features in images. For example, let us see some sample images from different lesion types from out data set as shown in [Fig 4.1].



**Fig 4.1: Category wise sample images**

In these images we can observe that important features in most of the images are circles and disk like shapes in these images. So, to differentiate between these lesion types our model should be able to focus on those circles and disk like shapes in the images. This ability to grasp the deeper features hidden in the images makes CNN very useful and puts it above other classification models. A typical CNN architecture is shown in [Fig 4.2].



**Fig 4.2**

This ability to grasp the deeper features hidden can be observed by using relevant filters in the convolution layer. After applying the filters the output of **convolution layer** is fed as input to **pooling layer** where the raw output is converted to predetermined features of the corresponding filters. In this way a raw image is converted to a list of features that we desired when applying filters. Then using these list of features we use normal neural network architecture to classify into one of the seven lesion types using **Fully Connected Layer**.

**4.2 Different Concepts Used**

This section deals with some concepts used over a basic CNN model to increase the prediction accuracy of the model.

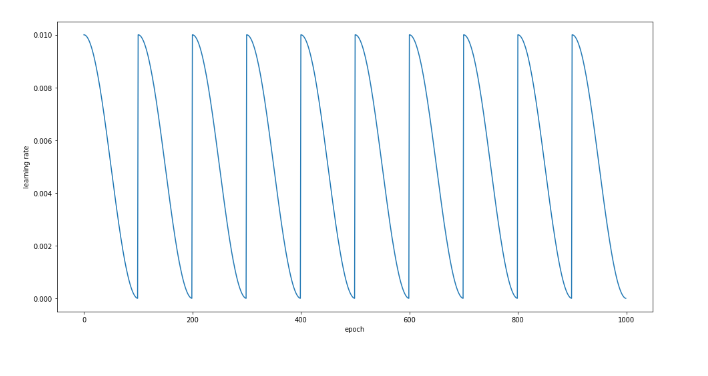
**1) Transfer Learning**

The idea behind the transfer learning is to use a CNN model which is pre trained on ImageNet data as lower layers of our model so that it can capture some generic features and

we fine tune the higher layers to our specific domain and redefine the last layer to output 7 values corresponding to different lesion types. After experimenting with different pre trained architectures for 5 epochs, I found that Resnet50 is giving better results. So I adopted that architecture as it gave the best validation accuracy upon training on our data. I also used Adam optimizer with weight decay to reduce overfitting.

**2) Cyclic Learning Rate Scheduling**

To improve the results and to make the model converge at global minimum instead of local minimum we have to periodically increase the learning instead of determining the optimal learning rate exponentially, I have used a Cyclic Learning Rate Scheduling. This varies the learning rate cyclically, which helps the model to escape several global minimum. It also eliminates the necessity to find an optimal learning rate manually. The Cyclic Learning rate used is shown in [Fig 4.3].

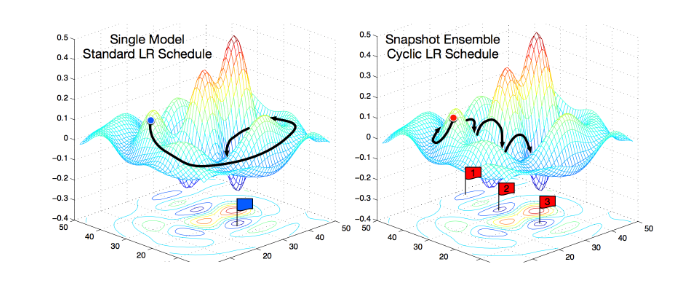


**Fig 4.3: Cyclic Learning Rate**

**3) Snapshot Ensemble**

Normal ensemble models are very powerful in improving the model performance. But, they also are computationally very expensive to separately train the model using different algorithms used in ensemble learning. So I have adopted a different ensemble technique known as **snapshot ensemble** with cyclic LR scheduling.

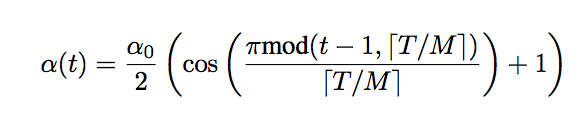
The main idea is to save the model parameters periodically during training. when a model converges to local minima during a cycle these parameters are saved and the learning rate is then increased to apply another model. In this way it allows us to gather an ensemble of models in a single training cycle.



**Fig 4.4**

[Fig 4.4] shows the differences between normal model and an snapshot ensemble model. In the later, we gradually save the snapshots at each local minimum and so we reach global minimum very fast. But in normal model we have to travel for a long time to reach global minimum. So this ensemble model helps us to reach global minimum in less epochs.

This can be implemented using the formula,



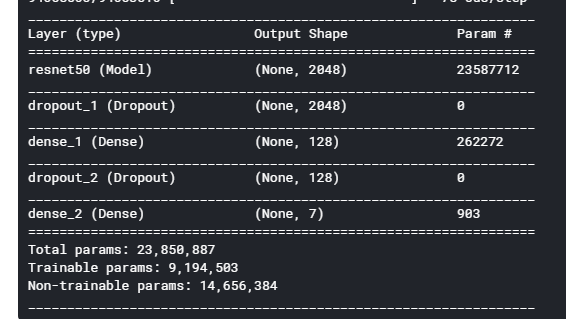
This formula gives the learning rate similar to that of [Fig 4.3]. So we can implement this model by simply applying this formula to get the desired learning rate.

**4.3 Model Building**

The model is built in python using Keras Sequential API, where you have to add one layer at a time.

First the Resnet50 architecture is added to the model. This is a pre trained architecture to capture generic features.

Then a dropout layer is added, dropout is a regularization method, here a portion of nodes are ignored randomly for each training sample. this makes network to learn features in a distributed way. This also improves the generalization and thereby reduces overfitting.



**Fig 4.5**

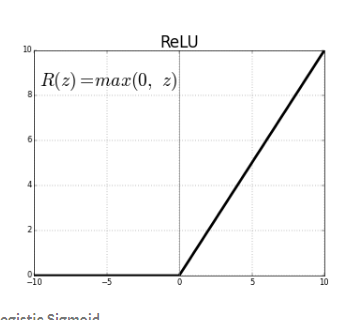
Then a dense layer with 128 nodes is added, this is a part of fully connected layer where different features from resnet are converted to give a output from 128 nodes. This is just an artificial neural network classifier.

Then again a dropout layer is added which is followed by the output dense layer with 7 nodes corresponding to 7 different lesion types, the net output gives the probability of each class.

My CNN architecture is,

Input => Resnet50 => Dropout => Dense => Dropout => Dense => Output

Complete details are shown in [Fig 4.5]. In both the dense layers 'relu' activation function is used. It adds non linearity to the network.



**Fig 4.6**

ReLU stands for rectified linear unit. It is the most commonly used activation function.

**4.4 Training the Model**

After building the model, it is compiled using Adam optimizer. when this model is trained with exponential learning rate reducer, test accuracy was risen to nearly 83% from 77% of normal CNN without resnet(Some other model). Then after using the Cyclic LR Scheduling and the **Snapshot Ensemble** techniques for training, test accuracy was risen to 92.71%.

This accuracy is obtained by trying different combinations of epochs and number of ensemble models. I have observed that during normal training procedure there is not a significant increase in the accuracy and even loss remained constant or even start increasing after **30 epochs**.

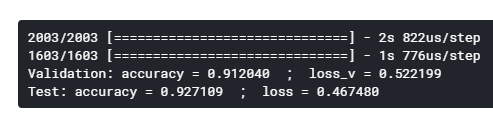
So to reduce the computation time, I tried different number of models and number of epochs combination so that, **(no. of models) x (no. of epochs per model** ) = **30**, such that total epochs for entire model remains 30. Finally among them the combination,**(No. of models = 3, and No. of epochs per model =10)** seems to be the optimal solution**. Max learning rate** is set to **0.001.** Learning rate decreases and then increases periodically to max learning rate. **Batch size** is set to **10**. Then model fitting is done with these parameters set on the training data, and validation data which is segmented earlier is used for validating the performance during training process. Once the training is done, Test data is used to evaluate the final performance of the model.

**Chapter 5**

**Results**

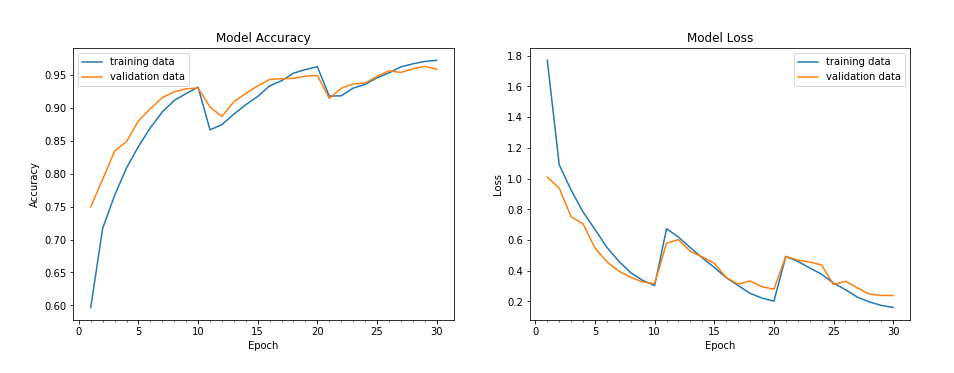
The proposed CNN model is used to classify the skin cancer type of the dataset that comprises the skin lesion images of humans. Images below shows the results of classification using test data and validation data.

The model gave the test accuracy of 92.71% and validation accuracy of 91.2% as shown in [Fig 5.1].



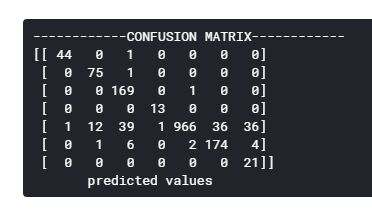
**Fig 5.1**

[Fig 5.2] shows the changes in accuracy and loss of training data and validation data during training process as the epochs progresses.



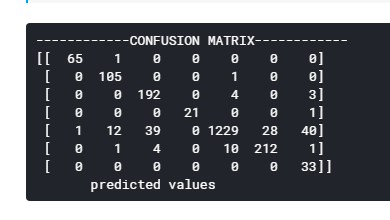
**Fig 5.2**

[Fig 5.3] shows the confusion matrix of validation data.



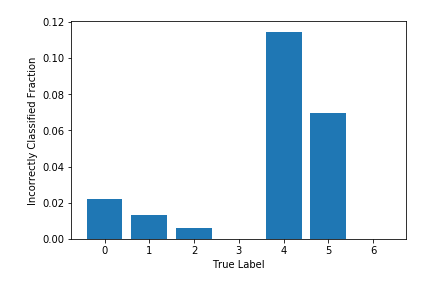
**Fig 5.3**

[Fig 5.4] shows the confusion matrix of test data.

****

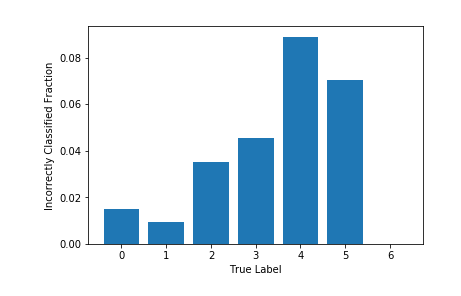
**Fig 5.4**

[Fig 5.5] shows the Incorrectly predicted fraction of each lesion type of validation data .



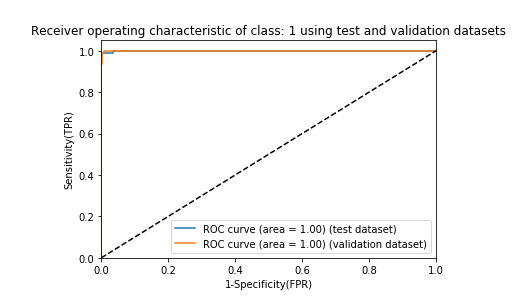
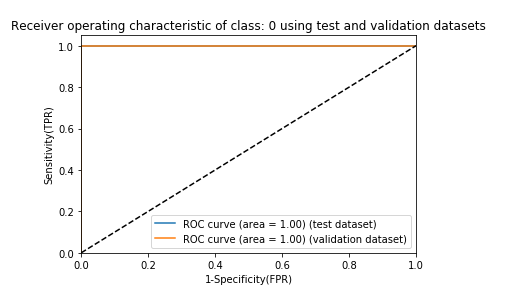
**Fig 5.5**

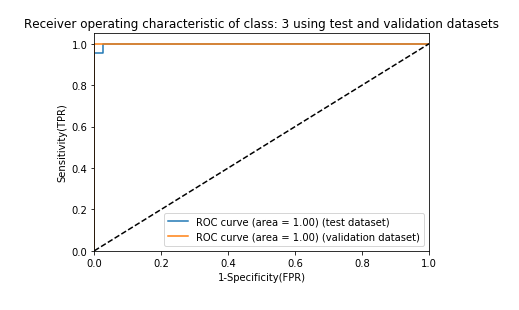
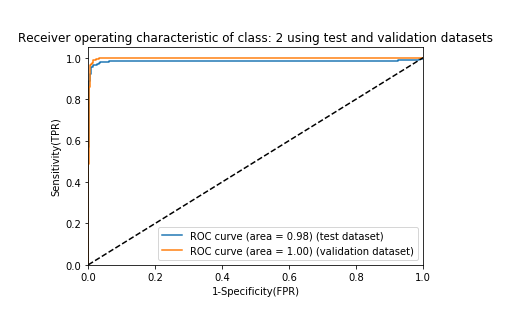
[Fig 5.6] shows the Incorrectly predicted fraction of each lesion type of test data .

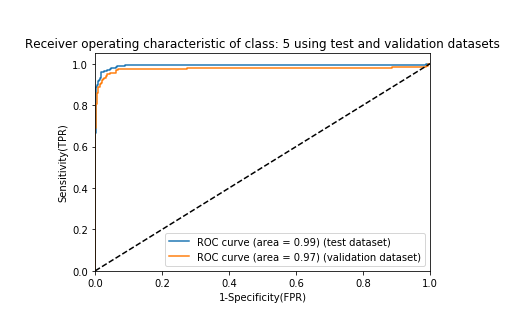
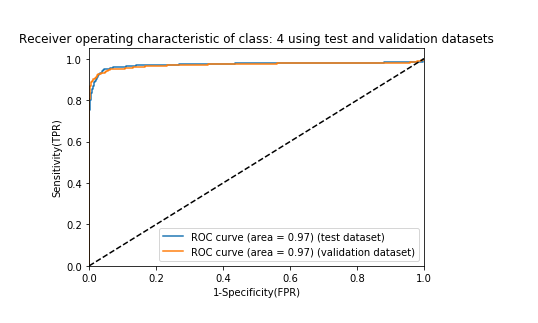


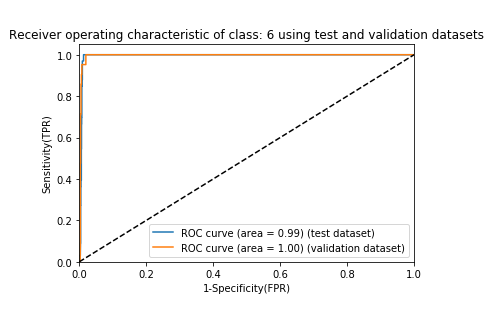
**Fig 5.6**

The following images shows the Receiver Operating Characteristic (ROC) curves of both test and validation data for each class.

****

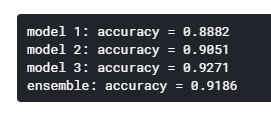
****

****

****

**Fig 5.7**

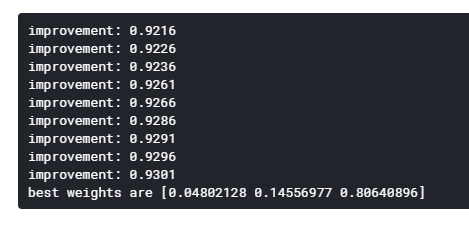
Snapshot Ensemble model used above have 3 models, when each snapshot saved after the model execution is evaluated using test data we get three different probabilities from three models. Using these three probabilities an **ensemble probability** can be calculated by taking the average of all three probabilities and the resultant class with highest probability is added to predicted data. Thus we can get an ensemble model with **ensemble accuracy**. That accuracy along with model accuracies are shown in [Fig 5.8].



**Fig 5.8**

In the above ensemble accuracy calculation we have taken the average of probability of all 3 models. Instead of average we can take **weighted average** to produce better results. But the problem is, how to chose the weights for different models. This can be done by initialising the weights randomly and then check for improvement, if there is any improvement then add new weights to **best weights**. If there is no improvement then increment the no improvement counter. We continue this process until the no improvement counter reaches certain limit and this limit is known as **patience**. Thus the final accuracy and final improved weights can be calculated.

By applying this method to our test data the final **ensemble accuracy is 93.01%**. The improvement process and the resultant best weights are shown in [Fig 5.9].



**Fig 5.9**

**Chapter 6**

**Future Work and Other Ideas**

This section contains the future work to be done and some ideas that are either not implemented or implemented improperly.

**Data Collection**

In many of the image classification projects correct results depends mostly on dataset. Our dataset contains 10,015 images, but most of those images are of only one class Melanocytic Nevi. It is important to collect a balanced data for better predictions.

Also our dataset has only seven different lesion types whereas in practical there are many more different types of skin cancer. One of the future work is to collect a better dataset with more balanced data with more lesion types.

**Other ensemble methods**

I also tried some other ensemble methods where model is trained separately for each algorithm. One such example is to try linear regression as one model, Decision tree as another algorithm, CNN as another algorithm and calculate the ensemble accuracy as mentioned in previous chapters.

The models I have tried are computationally much expensive with not much significant increase in accuracy. One future work is to try and find better ensemble models which produces better accuracy.

**Generative adversarial networks**

Data augmentation and class balancing plays an important role in model performance as seen previously. Besides classic image processing pre defined generative models can be used to improve performance. Some such models are, BAGAN for balancing and DAGAN for augmentation, etc,.

**Different Activation Functions in CNN**

In our CNN architecture we have used **ReLU** activation function. This is most commonly used activation function but we can try some novel activation functions available such as **MISH** activation function. It is a self regularised non-monotonic neural activation function given as,

𝑓(𝑥) = 𝑥 ⋅ 𝑡𝑎𝑛ℎ( ln(1 + 𝑒 𝑥 ) )

There are some other similar activation functions like **SWISH**. We can also use a customized activation function.

**Chapter 7**

**References**

[1] Arevalo J, Cruz-Roa A, Arias V, Romero E, Gonzalez FA. An unsupervised

feature learning framework for basal cell carcinoma image analysis. Artif

Intell Med 2015;64:131e45.

[2] Binder M, Steiner A, Schwarz M, Knollmayer S, Wolff K, Pehamberger H.

Application of an artificial neural network in epiluminescence microscopy

pattern analysis of pigmented skin lesions: a pilot study. Br J Dermatol

1994;13:460e5.

[3] Canziani A, Paszke A, Culurciello E. An analysis of deep neural network

models for practical applications, https://arxiv.org/abs/1605.07678; 2016

(accessed 14 April 2017).

[4]Codella N, Nguyen Q-B, Pankanti S, Gutman D, Helba B, Halpern A, et al.

Deep learning ensembles for melanoma recognition in dermoscopy images.

IBM J Res Dev 2017;61(4).

[5]Esteva A, Kuprel B, Novoa RA, Ko J, Swetter SM, Blau HM, et al. Dermatologist-

level classification of skin cancer with deep neural networks. Nature

2017;542(7639):115e8.

[6]Giotis I, Molders N, Land S, Biehl M, Jonkman MF, Petkov N. MED-NODE: a

computer-assisted melanoma diagnosis system using non-dermoscopic

images. Expert Syst Appl 2015;42:6578e85.

[7]Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks.

J Machine Learn Res 2011;15:315e23.

[8]Han SS, Park GH, Lim W, Kim MS, Na JI, Park I, et al. Deep neural networks

show an equivalent and often superior performance to dermatologists in

onychomycosis diagnosis: automatic construction of onychomycosis

datasets by region-based convolutional deep neural network. PLoS One

2018;13(1):e0191493.

[9]He K, Zhang X, Ren S, Sun J. Delving deep into rectifiers: surpassing humanlevel

performance on ImageNet classification, https://arxiv.org/abs/1502.

01852; 2015 (accessed 6 February 2015).

[10]Ioffe S, Szegedy C. Batch normalization: accelerating deep network training

by reducing internal covariate shift, https://arxiv.org/pdf/1502.03167v3.

pdf; 2015 (accessed 6 February 2015).

[11]Kato T, Suetake T, Sugiyama Y, Tabata N, Tagami H. Epidemiology and

prognosis of subungual melanoma in 34 Japanese patients. Br J Dermatol

1996;134:383e7.

[12]Kim GK, Del Rosso JQ, Bellew S. Skin cancer in asians: part 1:

nonmelanoma skin cancer. J Clin Aesthet Dermatol 2009;2(8):

39e42.

[13]Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep

convolutional neural networks. Adv Neural Inf Process Syst 2012;25:

1097e105.

[14]Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R.

Dropout: a simple way to prevent neural networks from overfitting. J Mach

Learn Res 2014;15(1):1929e58.

[15]Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going deeper

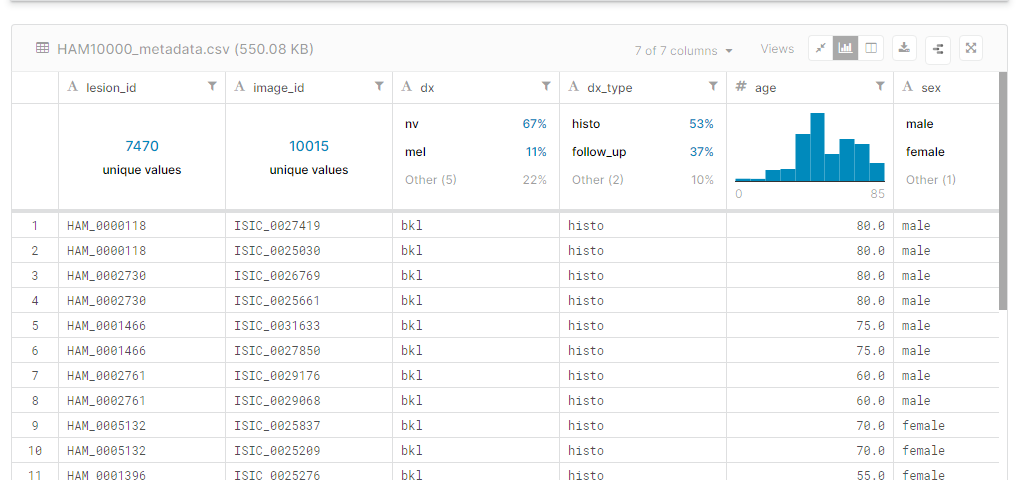
with convolutions, https://arxiv.org/abs/1409.4842; 2015 (accessed 17

September 2014).

**Appendix**

**Dataset Overview**

The following image shows the overview of raw metadata file used in the code.



This meta data file contains the image\_id field which has the image names of all the images present in the **HAM10000\_images\_part\_1** and **HAM10000\_images\_part\_2** directories present in the data.

**Important code snippets**

Our project is implemented in python notebook in kaggle using Keras and Tensorflow.The following are the 15 important code snippets used.

**1) Necessary libraries**

%matplotlib inline

import matplotlib.pyplot as plt

import numpy as np

import pandas as pd

import os

import itertools

import math

import keras

import seaborn as sns

from glob import glob

from PIL import Image

np.random.seed(123)

from sklearn.preprocessing import label\_binarize

from sklearn.metrics import confusion\_matrix

from sklearn.model\_selection import train\_test\_split

from keras.utils.np\_utils import to\_categorical

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D

from keras.callbacks import ReduceLROnPlateau

from keras.optimizers import Adam

from keras.preprocessing.image import ImageDataGenerator

from sklearn.model\_selection import train\_test\_split

from keras import backend as K

from keras.layers.normalization import BatchNormalization

from sklearn.metrics import roc\_curve, auc

# For Resnet

import pandas as pd

import numpy as np

from keras.models import Sequential

from keras import optimizers

from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D

from keras.applications import ResNet50

from keras import regularizers

# For snapshot Ensemble

from keras.callbacks import Callback

from keras import backend

from keras.models import load\_model

**2) Data Extraction**

skin\_directory = os.path.join('..', 'input/skin-cancer-mnist-ham10000')

# creating a directory for all images present with us and bringing them under same directory

image\_directory = {os.path.splitext(os.path.basename(x))[0]: x

for x in glob(os.path.join(skin\_directory, '\*', '\*.jpg'))}

# creating a directory to display the type of cancer with label values present in dataset

cancer\_type\_dict = {

'nv': 'melanocytic\_nevi',

'mel': 'melanoma',

'vasc': 'vascular\_lesions',

'df': 'dermatofibroma',

'bkl': 'benign\_keratosis\_like\_lesions ',

'bcc': 'basal\_cell\_carcinoma',

'akiec': 'actinic\_keratoses'

}

# creating data frame

skin\_df = pd.read\_csv(os.path.join(skin\_directory, 'HAM10000\_metadata.csv'))

# Creating New Columns for better readability

skin\_df['path'] = skin\_df['image\_id'].map(image\_directory.get)

skin\_df['cell\_type'] = skin\_df['dx'].map(cancer\_type\_dict.get)

skin\_df['cell\_type\_idx'] = pd.Categorical(skin\_df['cell\_type']).codes

**3) Resizing images**

skin\_df['image'] = skin\_df['path'].map(lambda x: np.asarray(Image.open(x).resize((100,75))))

**4) Data Balancing**

# Copy fewer class to balance the number of 7 classes

data\_aug\_rate = [15,10,5,50,0,5,40]

for i in range(7):

if data\_aug\_rate[i]:

skin\_df=skin\_df.append([skin\_df.loc[skin\_df['cell\_type\_idx'] == i,:]]\*(data\_aug\_rate[i]-1), ignore\_index=True)

skin\_df['cell\_type'].value\_counts()

**5) Data splitting**

x\_train\_o, x\_test\_o, y\_train\_o, y\_test\_o = train\_test\_split(features, target, test\_size=0.20,random\_state=3)

**6) Data Normalization**

x\_train = np.asarray(x\_train\_o['image'].tolist())

x\_test = np.asarray(x\_test\_o['image'].tolist())

x\_train\_mean = np.mean(x\_train)

x\_train\_std = np.std(x\_train)

x\_test\_mean = np.mean(x\_test)

x\_test\_std = np.std(x\_test)

x\_train = (x\_train - x\_train\_mean)/x\_train\_std

x\_test = (x\_test - x\_test\_mean)/x\_test\_std

**7) Model Building**

model = Sequential()

num\_labels = 7

base\_model = ResNet50(include\_top=False, input\_shape=(75, 100, 3),pooling = 'avg', weights = 'imagenet')

model = Sequential()

model.add(base\_model)

model.add(Dropout(0.5))

model.add(Dense(128, activation="relu",kernel\_regularizer=regularizers.l2(0.02)))

model.add(Dropout(0.5))

model.add(Dense(num\_labels, activation = 'softmax',kernel\_regularizer=regularizers.l2(0.02)))

for layer in base\_model.layers:

layer.trainable = False

for layer in base\_model.layers[-22:]:

layer.trainable = True

model.summary()

**8) Data Augmentation**

# With data augmentation to prevent overfitting

datagen = ImageDataGenerator(

featurewise\_center=False, # set input mean to 0 over the dataset

samplewise\_center=False, # set each sample mean to 0

featurewise\_std\_normalization=False, # divide inputs by std of the dataset

samplewise\_std\_normalization=False, # divide each input by its std

zca\_whitening=False, # apply ZCA whitening

rotation\_range=10, # randomly rotate images in the range (degrees, 0 to 180)

zoom\_range = 0.1, # Randomly zoom image

width\_shift\_range=0.1, # randomly shift images horizontally (fraction of total width)

height\_shift\_range=0.1, # randomly shift images vertically (fraction of total height)

horizontal\_flip=False, # randomly flip images

vertical\_flip=False) # randomly flip images

**9) Model Fitting**

print(len(x\_validate))

print(len(y\_validate))

se\_callback = SnapshotEnsemble(n\_models=3, n\_epochs\_per\_model=10

, lr\_max=.001)

history = model.fit\_generator(

datagen.flow(x\_train, y\_train, batch\_size=10),

steps\_per\_epoch=len(x\_train) / 10,

epochs=se\_callback.n\_epochs\_total,

verbose=1,

callbacks=[se\_callback],

validation\_data=(x\_validate, y\_validate)

)

**10) Evaluating accuracy of test and validation data**

loss, accuracy = model.evaluate(x\_test, y\_test, verbose=1)

loss\_v, accuracy\_v = model.evaluate(x\_validate, y\_validate, verbose=1)

print("Validation: accuracy = %f ; loss\_v = %f" % (accuracy\_v, loss\_v))

print("Test: accuracy = %f ; loss = %f" % (accuracy, loss))

model.save("model.h5")

**11) Plotting accuracy and loss graps**

# plotting accuracy and loss of training data and validation data with progression in epochs during training process

model\_history = history

fig, plots = plt.subplots(1,2,figsize=(15,5))

plots[0].plot(range(1,len(model\_history.history['acc'])+1),model\_history.history['acc'])

plots[0].plot(range(1,len(model\_history.history['val\_acc'])+1),model\_history.history['val\_acc'])

plots[0].set\_title('Model Accuracy')

plots[0].set\_ylabel('Accuracy')

plots[0].set\_xlabel('Epoch')

plots[0].set\_xticks(np.arange(1,len(model\_history.history['acc'])+1),len(model\_history.history['acc'])/10)

plots[0].legend(['training data', 'validation data'], loc='upper left')

plots[1].plot(range(1,len(model\_history.history['loss'])+1),model\_history.history['loss'])

plots[1].plot(range(1,len(model\_history.history['val\_loss'])+1),model\_history.history['val\_loss'])

plots[1].set\_title('Model Loss')

plots[1].set\_ylabel('Loss')

plots[1].set\_xlabel('Epoch')

plots[1].set\_xticks(np.arange(1,len(model\_history.history['loss'])+1),len(model\_history.history['loss'])/10)

plots[1].legend(['training data', 'validation data'], loc='upper right')

plt.show()

**12) Printing Confusion Matrix**

# Predicted values

Y\_pred = model.predict(x\_validate)

Y\_pred\_test = model.predict(x\_test)

# Converting to one hot vectors

Y\_pred\_classes = np.argmax(Y\_pred,axis = 1)

Y\_pred\_test\_classes = np.argmax(Y\_pred\_test,axis = 1)

Y\_true = np.argmax(y\_test,axis = 1)

# confusion matrix calculation

confusion\_mtx = confusion\_matrix(Y\_true, Y\_pred\_test\_classes)

#printing confusion matrix

print('------------CONFUSION MATRIX------------')

print(confusion\_mtx)

print(' predicted values ')

**13) Plotting ROC curves**

y\_pred\_test = model.predict(x\_test)

y\_pred\_val = model.predict(x\_validate)

num\_classes=7

fpr\_val = dict()

tpr\_val = dict()

roc\_auc\_val = dict()

fpr\_test = dict()

tpr\_test= dict()

roc\_auc\_test = dict()

for i in range(num\_classes):

fpr\_test[i], tpr\_test[i], \_ = roc\_curve(y\_test[:, i], y\_pred\_test[:, i])

roc\_auc\_test[i] = auc(fpr\_test[i], tpr\_test[i])

fpr\_val[i], tpr\_val[i], \_ = roc\_curve(y\_validate[:, i], y\_pred\_val[:, i])

roc\_auc\_val[i] = auc(fpr\_val[i], tpr\_val[i])

# Plot of a ROC curve for a specific class

for i in range(num\_classes):

plt.figure()

plt.plot(fpr\_test[i], tpr\_test[i], label='ROC curve (area = %0.2f) (test dataset)' % roc\_auc\_test[i])

plt.plot(fpr\_val[i], tpr\_val[i], label='ROC curve (area = %0.2f) (validation dataset)' % roc\_auc\_val[i])

plt.plot([0, 1], [0, 1], 'k--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('1-Specificity(FPR)')

plt.ylabel('Sensitivity(TPR)')

plt.title('Receiver operating characteristic of class: '+ str(i) + ' using test and validation datasets' )

plt.legend(loc="lower right")

plt.show()

**14) Ensemble Accuracy Calculation**

from sklearn.metrics import accuracy\_score

# makes prediction according to given models and given weights

def predict(models, data, weights=None):

if weights is None:

# default weights provide voting equality

weights = [1 / (len(models))] \* len(models)

pred = np.zeros((data.shape[0], 7))

for i, model in enumerate(models):

pred += model.predict(data) \* weights[i]

return pred

# returns accuracy for given predictions

def evaluate(preds, weights=None):

if weights is None:

weights = [1 / len(preds)] \* len(preds)

y\_pred = np.zeros((y\_test.shape[0], 7))

for i, pred in enumerate(preds):

y\_pred += pred \* weights[i]

y\_pred = np.argmax(y\_pred, axis=1)

y\_true = np.argmax(y\_test, axis=1)

return accuracy\_score(y\_true, y\_pred)

# load list of snapshots

models = se\_callback.load\_ensemble()

# precalculated predictions of all models

preds = []

# evaluate every model as single

for i, model in enumerate(models):

pred = predict([model], x\_test)

preds.append(pred)

score = evaluate([pred])

print(f'model {i + 1}: accuracy = {score:.4f}')

# evaluate ensemble (with voting equality)

ensemble\_score = evaluate(preds)

print(f'ensemble: accuracy = {ensemble\_score:.4f}')

**15) Ensemble Accuracy Improvement and Best Weights calculation**

best\_score = ensemble\_score

best\_weights = None

no\_improvements = 0

while no\_improvements < 5000: #patience

# generate normalized weights

new\_weights = np.random.uniform(size=(len(models), ))

new\_weights /= new\_weights.sum()

# get the score without predicting again

new\_score = evaluate(preds, new\_weights)

# check (and save)

if new\_score > best\_score:

no\_improvements = 0

best\_score = new\_score

best\_weights = new\_weights

print(f'improvement: {best\_score:.4f}')

else:

no\_improvements += 1

print(f'best weights are {best\_weights}')