# 

# Automated Diagnosis of Skin Lesions

Project Report

**Group Project: CMPE 255 - Data Mining**

**Team Members**

**​Krishna Sai Siva Bojja (012435592)**

**Nasrajan Jalin (013784420)**

**Shalu Rani (012565046)**

# 

# Abstract

Since the advent of Data Mining technologies, every organization has started storing enormous amount of data. The volume of data generated from various medical diagnostic devices is huge and most of them are images too. Applying appropriate software technologies, especially data mining techniques, can bring in more accuracy and speed to different diagnoses and avoid human errors. This project intends to automate the classification of different skin lesions and patterns in the initial stage itself so that the patients get early treatment. Since melanoma ( a type of skin cancer) is incurable if detected in later stages, identification and treatment in early stage is recommended.

# 1. Introduction

The HAM10000 dataset, is a collection of dermatoscopic images of common pigmented skin lesions. The images are one of the seven diagnostic areas - Actinic keratoses and intraepithelial carcinoma / Bowen's disease (akiec), basal cell carcinoma (bcc), benign keratosis-like lesions (solar lentigines / seborrheic keratoses and lichen-planus like keratosis, bkl), dermatofibroma (df), melanoma (mel), melanocytic nevi (nv) and vascular lesions (angiomas, angiokeratomas, pyogenic granulomas and hemorrhage, vasc).

This project aims to build a classification model which accurately classifies each given image to one of the seven classes.

## Motivation

Skin cancer is one of the most common types of cancers in the world. Let us look at

some statistics:

* More people are diagnosed in US with skin cancer compared to all other cancers combined.
* 1 in 5 Americans develop skin cancer by the age of 70.

It is a well known fact that, most forms of cancer become easy to treat if diagnosed

early. If there is an automated model built from historical data, which can give hints to

doctors on the possibilities of a skin cancer from the image in front of them, it can save

many lives. Such an application will especially be a life saver in developing countries, where doctor’s intuition weigh more than the diagnostic parameters.

## b. Objective

The project aims to build a classification model, which is trained from historical data and

will classify photos of skin lesions to a class of skin disease with more than 70%

Accuracy.

## c. Market Study

Previous works related to the classification of cancerous images were quite limited to

two classes of melanocytic images. This was mostly due to the unavailability of large

amount of data. Most of the historical medical data are protected with

copyrights and not publically available, for obvious privacy reasons. Another issue with

current data sets are lack of diversity.

# 2. System Design & Implementation details

## Algorithm(s) considered/selected

The image classifier was built using Convolutional Neural Network (CNN). CNN is a class of deep, forward-feed artificial neural networks, most commonly applied to analyzing visual imagery. CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. Sequential model was used in order to to build linear stack of layers. In this model, 4 convolution layers , 2 pooling layers, 3 dropout layers, 1

flatten layer and 2 dense layers were used. Number of layers was selected after experimenting with various combinations.

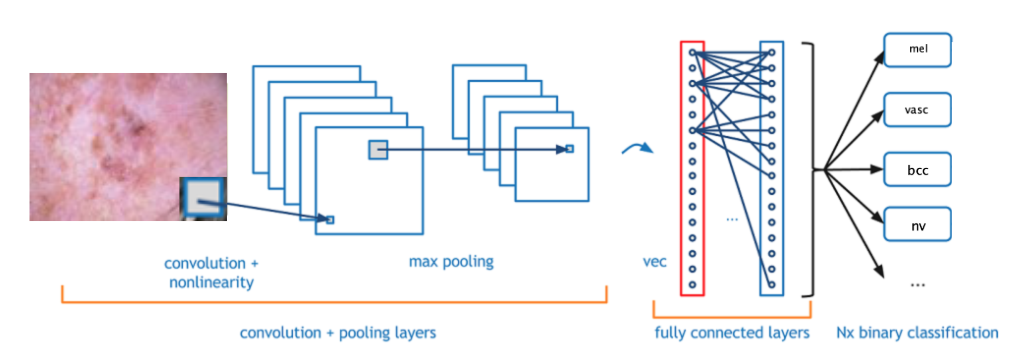


Figure above shows the different layers

Below is the pseudocode of the algorithm used for implementation:

**Step1:** Image resizing was done using Python’s Image library.

**Step2:** Metadata file was read using pandas, which contained image’s metadata

information like: lesionID, imageID, category of skin cancer, affected body part, age &

sex of the person etc.

**Step3:** Data preprocessing was done:

* It was observed that data was imbalanced among different class categories,

therefore resampling was done to generate and use equal number of samples for each category for training and validation.

* Null values in age field were replaced by using pandas fillna function.
* Actual images were mapped to the metadata information using a new field “path”

**Step4:** Python Image Library was used for image processing.

**Step5:** Python’s sklearn library was used for splitting the dataset into training and testing

Samples to avoid overfitting. The split ratio between training and testing data was 80:20

in percentage.

Numpy library was used for perceiving the image data in matrix format.

**Step6:** The training and testing data were normalized by subtracting mean and dividing

by standard deviation (normalization by standard deviation).

**Step7:** Training data was further split into 90:10 ratio where 90% data was training and

10% data was validation data.

**Step8:** Keras sequential model was used for modeling. A total of 12 core layers were

added for modeling which included Conv2D layer, MaxPool2D layer, Dropout layer,

Flatten layer and Dense layer. The specific parameters for each layer was set explicitly.

**Step9:** ImageDataGenerator API in Keras was used for image preprocessing and for

creating augmented image generator with various parameters like random rotations,

shifts, flips etc.

**Step10:** Current training data was fitted into the image generator and was further fitted

into the model.

**Step11:** The model was compiled using SGD optimizer, loss function as

categorical\_crossentropy and the metrics as accuracy. The optimizer Adam was also

used in some models.

**Step12:** The training dataset was fitted in model for different configurations of sample

size and number of epochs.

**Step13:** For taking decision on the model selection, loss and accuracy for each model

was computed using evaluate function. The model resulting in least loss and maximum

accuracy was considered at the end.

**Step14:** All the models with different configurations were saved and tested for some

sample data. Various graphs were also plotted for comparative analysis and decisions.

## b. Technologies & Tools used (and why)

Programming language : Python

Tools used: Jupyter notebook

Version Control: Bitbucket (Git)

Since our team was comfortable in programming in python and everyone shared this

skillset, we opted to use python for programming. Also, we had been mostly using

jupyter notebook for running python code and capturing visualizations, therefore, we

chose jupyter notebook for creating visualisations.

## c. Architecture related decisions

As part of data mining course, we learnt about the feed forward neural network

architecture. In NN, the information propagates in forward direction with the help of

bias units(weights) and activation functions between the input, hidden and output layers,

the neurons learn the data/image.

Since convolutional neural networks(CNN) are widely used for image classification, we

opted to use it. The advantage is that it can handle redundant attributes as well

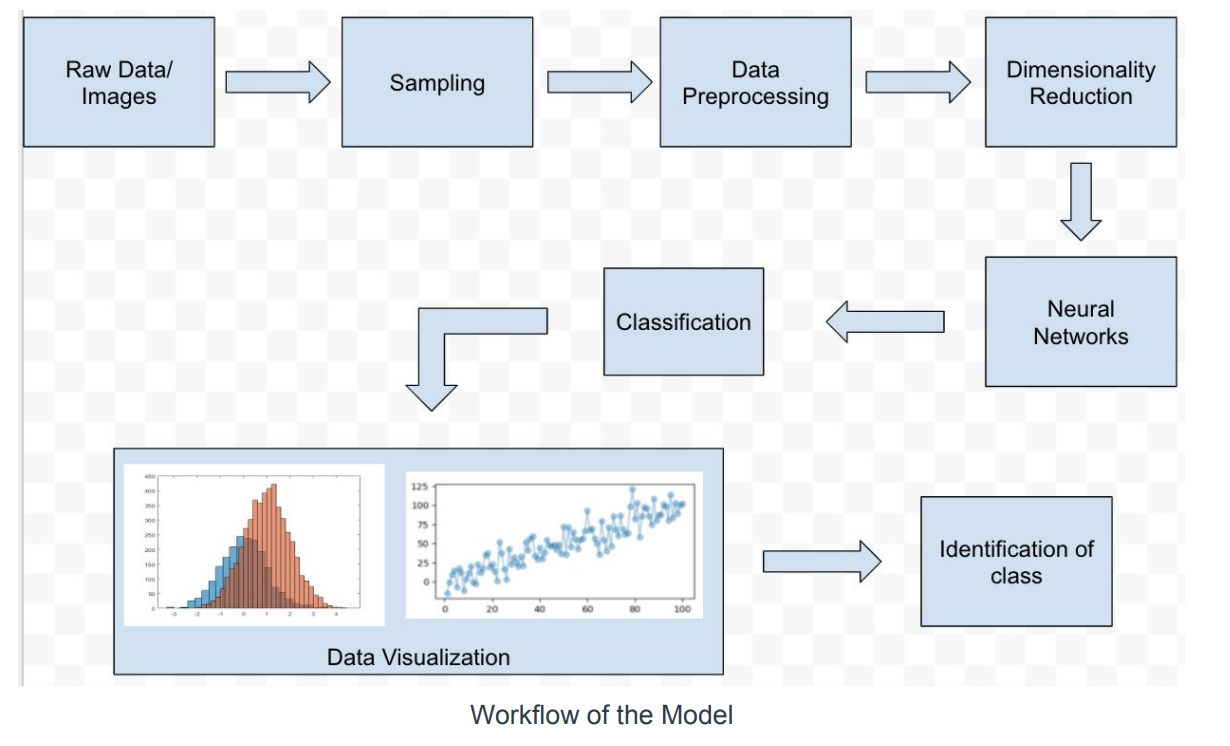
considering the fact that the weights are automatically learnt.

The number of layers and epochs were chosen after experimenting the model with

various combinations of values. The final model chosen was : Sequential, with 12 layers

and 75 epochs.

## d. System Architecture



## e. Component details

Data preprocessing: Image resizing using separate python code was done.

Model: Image classification model was built using Convolutional Neural Networks using

python in jupyter notebook.

Results: The results were plotted in Jupyter notebook.

## f. Use case

Use Case: Detecting of skin lesions into one of the 7 categories of skin

cancer. A doctor can use this model for identifying the category in which the skin cancer

lesion falls. Based on the result, appropriate consultation and care can be provided to

the patient.

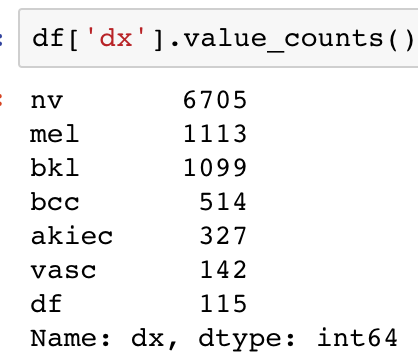
# 3. Experiments / Proof of concept evaluation

1. Dataset(s) used (name, source, type of data, size of data, # of instances/statistics, any preprocessing performed etc.)

The dataset used was MNIST multi-source dermoscopic images of pigmented lesions from Kaggle - https://www.kaggle.com/kmader/skin-cancer-mnist-ham10000/version/2 Dataset included more than 10015 images of lesions. It is a HAM10000 (Human Against Machine with 10000 training images) data set. The primary source is <https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T>

The total file size was around 2.57GB. Each image was of size 600 x 450 pixels

The class-wise distribution of data has been given below:



**Preprocessing:**

1.**Dimensionality Reduction:** The images were resized into 100 x 75 pixels.

2. **Resampling:** In order to correct the unbalanced distribution of data shown

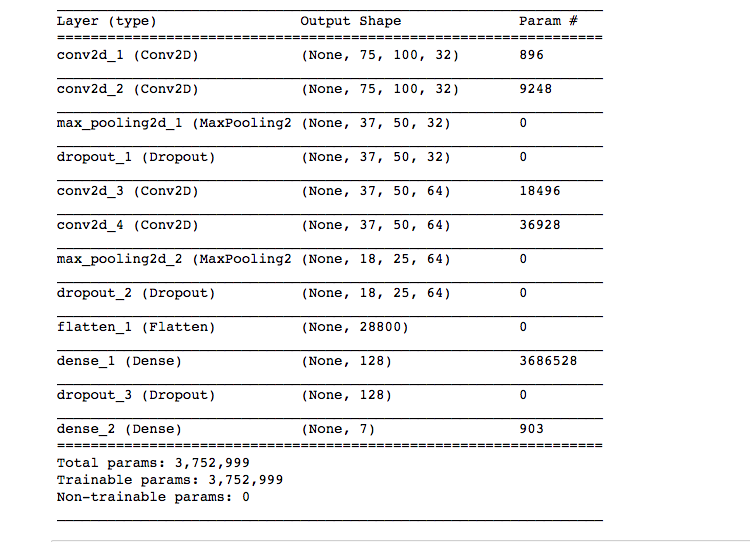
above, the images were resampled using sklearn utils.

3. **Normalization:** After converting each image to a numpy array, they were

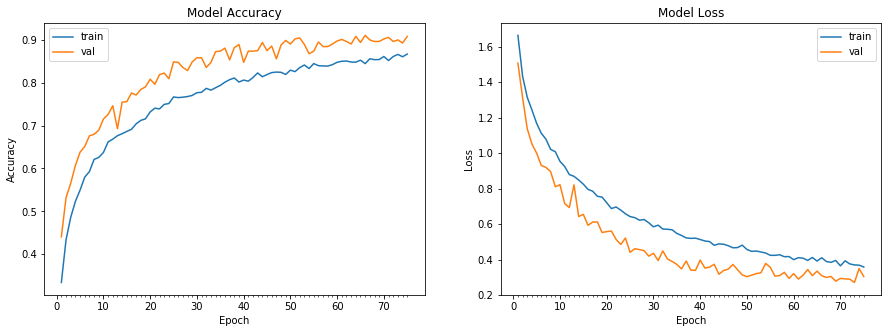
normalized using normalization by standard deviation.

1. Methodology followed

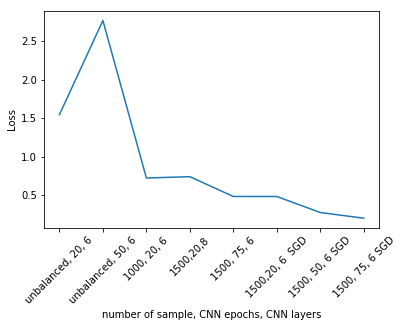
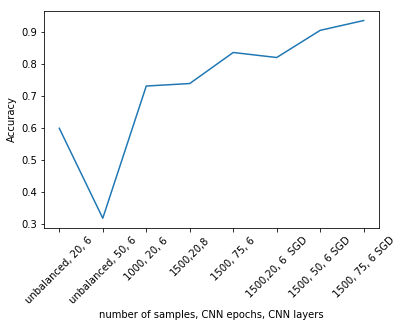
Convolutional Neural Network with 75 epochs and 12 keras layers. The size of training set was 80%. The training set was again split into training and validation data to prevent overfitting. The screenshot of the model layers have been shown below:



1. Graphical Plots



Model accuracy and loss graph for 1500 samples of each category and 75 epochs.



Test accuracy and loss comparisons with various configurations - number of samples,

epochs and optimizer used. The default optimizer for initial runs was Adam. With SGD,

improved results were observed.

## d. Analysis of results

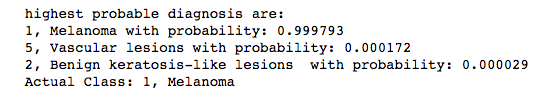
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.No | Samples per class | Number of epochs | Other changes | Accuracy | Loss |
| 1 | unbalanced | 20 | Layers=12 | 0.5993 | 1.5436 |
| 2 | unbalanced | 50 | Layers=12 | 0.3179 | 2.7663 |
| 3 | 1000 | 20 | Layers=12 | 0.7314 | 0.7213 |
| 4 | 1000 | 50 | Layers=12 | 0.7621 | 0.6351 |
| 5 | 1500 | 20 | Layers=12 | 0.7950 | 0.5490 |
| 6 | 1500 | 20 | Layers=16 | 0.7393 | 0.7383 |
| 7 | 1500 | 50 | Layers=12 |  |  |
| 8 | 1500 | 75 | Layers=12 | 0.8364 | 0.4813 |
| 9 | 1500 | 20 | Layers=12, Optimizer=SGD | 0.8207 | 0.4810 |
| 10 | 1500 | 50 | Layers=12, Optimizer=SGD | 0.9057 | 0.2730 |
| 11 | 1500 | 75 | Layers=12, Optimizer=SGD | 0.9364 | 0.1991 |

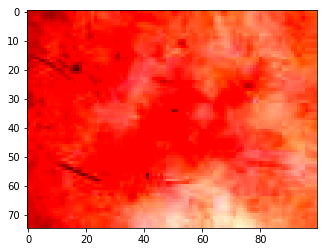
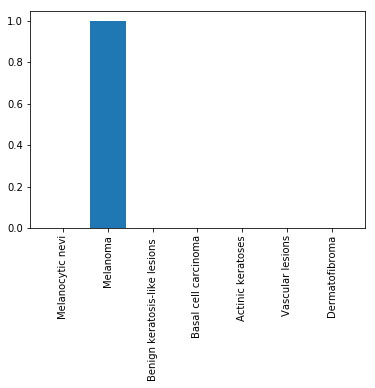
The CNN Model was run with 20, 50 and 75 epochs for different samples of data and

these were the general outcome:

* For unbalanced data, number of epochs in CNN didn’t have any impact. In fact, increase in number of epochs showed a steep decrease in accuracy and steep increase in loss.
* When balanced data was used, number of samples directly influenced the accuracy, inversely affected the loss. For more number of samples, accuracy was more and loss was less.
* When balanced data was used, for the same number of samples more epochs generated better accuracy and less loss.
* Adding SGD Optimizer improved the accuracy further. A combination of 1500 samples, 75 epochs along with SGD optimizer generated a model accuracy of 0.885 and a test accuracy of 0.93642.

Given below is the test generated for one image from the test set.



# 4. Discussion & Conclusions

## Decisions Made

**CNN model** : Since the dataset is medical image data, ensuring that the accuracy is maximal and los is minimal was important. Therefore, CNN model was chosen.

**Final Model Configuration**: 1500 samples, 75 epochs, SGD Optimizer , 12 layers

It generated the best accuracy and minimal loss out of experimental analysis.

**Final Model’s Accuracy:** 0.9364

**Final Model’s Loss:** 0.1991

## b. Difficulties faced

* + **Installation** **and Setup**: There were some package dependencies which had to be resolved for the program to run on Mac systems. Python 3.7 does not support Tensorflow. The version conflicts had to be resolved for the program to run. (Detailed setup instructions provided as a separate document).
  + **Imbalanced data**: We had a huge dataset of 10000 images, where each image belonged to one of seven classes namely nv, mel, bkl, akiec, vasc, bcc, df. The number of images for each class varied by large margin.
  + **Null Values**: Almost all huge datasets are inevitable of having null values. In this dataset the column age had 57 null values.

## c. Things that worked

* + Transforming to Balanced data: Most of the datasets used for classification do not have exactly the same number of instances in all their classes. The original data we had was highly imbalanced. The class nv had around 6705 images but df class had only 115 images. So we balanced the data by using the module resample from sklearn.utils library. We downsampled all the classes having images more than 1000 and upsampled all the classes having images less than 1000. So, the resultant dataset had 1000 images for each class and had overall dataset of 7000 images.
  + Model Creation using CNN and achieving accuracy above 80%.

## d. Conclusion

We built a Convolution Neural Network model that can predict the images with an

accuracy of 0.8852, using keras. The loss was observed to be 0.3777 considering 1500

samples of each image category and 75 epochs.

Considering the importance of a dataset and application of this kind in real world, there

is still a long way to go. We hope that this project can trigger productive discussions in

that direction.

# 5. Project Plan / Task Distribution

|  |  |  |
| --- | --- | --- |
| **Task** | **Assigned to** | **Done by** |
| Initial Research | Nasrajan | Nasrajan |
| Examine Data set | Krishna | Krishna |
| Data Preprocessing - Dimensionality Reduction | Krishna | Krishna |
| Data Preprocessing - Resampling | Krishna | Krishna |
| Data Preprocessing - Data cleansing | Shalu | Shalu |
| Data Normalization - Standard deviation | Krishna | Krishna |
| Data Optimization - SGD | Shalu | Shalu |
| CNN Model - Creation | Shalu | Shalu |
| CNN Model - various configurations (layers, epochs, hyper parameters) | Shalu | Shalu |
| Testing and comparison | Nasrajan | Nasrajan |
| Plots | Nasrajan | Nasrajan |
| Output Analysis | Nasrajan | Nasrajan |

# 6. References

* [*https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T*](https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T)
* [*https://www.skincancer.org/skin-cancer-information/skin-cancer-facts*](https://www.skincancer.org/skin-cancer-information/skin-cancer-facts)
* [*https://en.wikipedia.org/wiki/Convolutional\_neural\_network*](https://en.wikipedia.org/wiki/Convolutional_neural_network)
* [*https://keras.io/optimizers/*](https://keras.io/optimizers/)