Final Project Report
Skin Cancer Screening

Authors:

Harshvardhan Singh
School of Information, University of Arizona, Tucson, Az

Nassim Sbai
School of Information, University of Arizona, Tucson, Az

Content:

Introduction:

For a long time skin cancer has been a major cause of heath concern among people, with skin cancer being the most common cancer in the United States. Dermatologists are usually overbooked and seeing one sometimes takes a couple weeks. This is partially due to the fact that the diagnosis is slow in some situations. Our objective was to use our knowledge in machine learning and deep learning to develop an algorithm that will serve both the dermatologists and their patients by increasing the efficiency of the diagnosis. Our algorithm will allow dermatologists to detect cancerous skin conditions and know exactly to which type it is. By speeding up this process, the patients will be cured faster and a lot of complications will be prevented. We used the HAM1000 dataset to train our model, the dataset came with a collection of over 10,000 images.[1]

HAM 1000

[1] https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T

Skin Lesion:

Our dataset contained seven type of skin cancers, these include: Actinic keratoses and intraepithelial carcinoma / Bowen's disease, basal cell carcinoma, benign keratosis-like lesions (solar lentigines / seborrheic keratoses and lichen-planus like keratosis, dermatofibroma, melanoma, melanocytic nevi and vascular lesions (angiomas, angiokeratomas, pyogenic granulomas and hemorrhage. Actinic keratoses is a skin condition that shows up in the form of a rough, scaly patch that develops from years of sun exposure. Bowen's disease is an early sign of skin cancer that can be treatable using excision which cuts out the tumor. All the skin conditions mentioned in our dataset are cancerous and impact the skin priorly. Many of these conditions are caused by long hours of sun exposure over the years, some are solely caused by genetics. These seven types of skin cancers served as the labels in our model and our main goal was to develop an algorithm that is able to classify the picture of skin leisure or bruise and assign to a type of skin condition. While this is very effective, our algorithm assumes that any image of a skin leisure is malignant which is not the case. This is because our dataset only included malignant skin conditions that are cancerous. The model we developed will allow dermatologists to speed up their diagnosis and automatically identify the type of skin condition a leisure belongs to if it is cancerous.

Data Description:

Our dataset consists of over 10,000 image data, and each image has got a cereal number which acts as the image ID. The image names are stored in the metadata as the image ID, and has a unique lesion ID, age, sex, localization, but most importantly dx (pigmented lesion name), and lesion type. This is a multiclass dataset, 7 different classes which are-

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

Auto Keras to analyze best performing model:

Auto Keras is an open source library that allows us to discover the best performing algorithm for a given dataset. It is highly applied for Neural Networks where it allows us to select the best possible model based on the model architecture and the hyperparameters. It uses Keras models via the TensorFlow tf.keras API. The big advantage of this library is that it allows us to automatically find top performing models for both classification and regression with a very easy interface. For our project, we used Auto Keras to automatically select the appropriate model for our dataset. The autokeras trained 25 models for 20 epochs each, and the sequential model turned out to be the best model. By choosing this model, it allowed us to implement a model that produced a 70% prediction accuracy, this means that 70% of the observations in the test set were correctly classified by our model.

CNN (Convolutional Neural Network):

CNN is a deep learning algorithm that takes an input image, assigns different weights and biases to various characteristics of the image which allows it to recognize an image and classify it adequately. While neural networks are viewed as challenging to implement, with diverse Python libraries that we will describe in detail in the next paragraph, we were able to to develop a CNN algorithm with the adequate number of layers and proper architecture.

Code:

We have trained a CNN sequential model with keras to classify seven types of skin lesions. This is a multi-class classification model, and these are the seven skin lesions that are either cancerous already like the Melanoma which is the most serious type of skin cancer, or have a really high chance of turning cancerous like the Actinic keratoses, or is a benign skin lesion like the Benign keratosis-like lesions.

We execute our code in the following steps:

- Import libraries
- Load the dataset
- Preprocessing
- Analyzing the distribution of data
- Balance data
- Splitting of the data into training and test set
- Define and train the model
- Model analysis and visualization

1) Import Libraries

We first imported the required libraries including matplotlib which we will be using for visualization, Numpy for arrays and matrices, pandas for data manipulation and analysis, glob to get the pathnames, PIL, seaborn, sklearn, and keras.

In [2]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import os
from glob import glob
from PIL import Image
import seaborn as sns
```

In [3]:

```
np.random.seed(42)
from keras.utils.np_utils import to_categorical
from sklearn.model_selection import train_test_split
from scipy import stats
from sklearn.preprocessing import LabelEncoder
```

```
#import more libraries

from sklearn.metrics import confusion_matrix
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, BatchNormalization
num_classes = 7
```

2) Load the dataset

We load our dataset which is a csv file, and it's actually the metadata of our image dataset. We have got the image ID, which works as the identifiers for our image dataset, and dx is one of our seven classes. Other than that we have got age, sex, localization meaning which part of the body is the skin lesion located at.

In [4]:

```
#reading the metadata csv file
skin_df = pd.read_csv('D:\Dataset\HAM10000_metadata')
```

HAM10000_metadata

lesion_id	image_id	dx	dx_type	age	sex	localization
HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp
HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp
HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp
HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp
HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear
HAM_0001466	ISIC_0027850	bkl	histo	75.0	male	ear
HAM_0002761	ISIC_0029176	bkl	histo	60.0	male	face
HAM_0002761	ISIC_0029068	bkl	histo	60.0	male	face

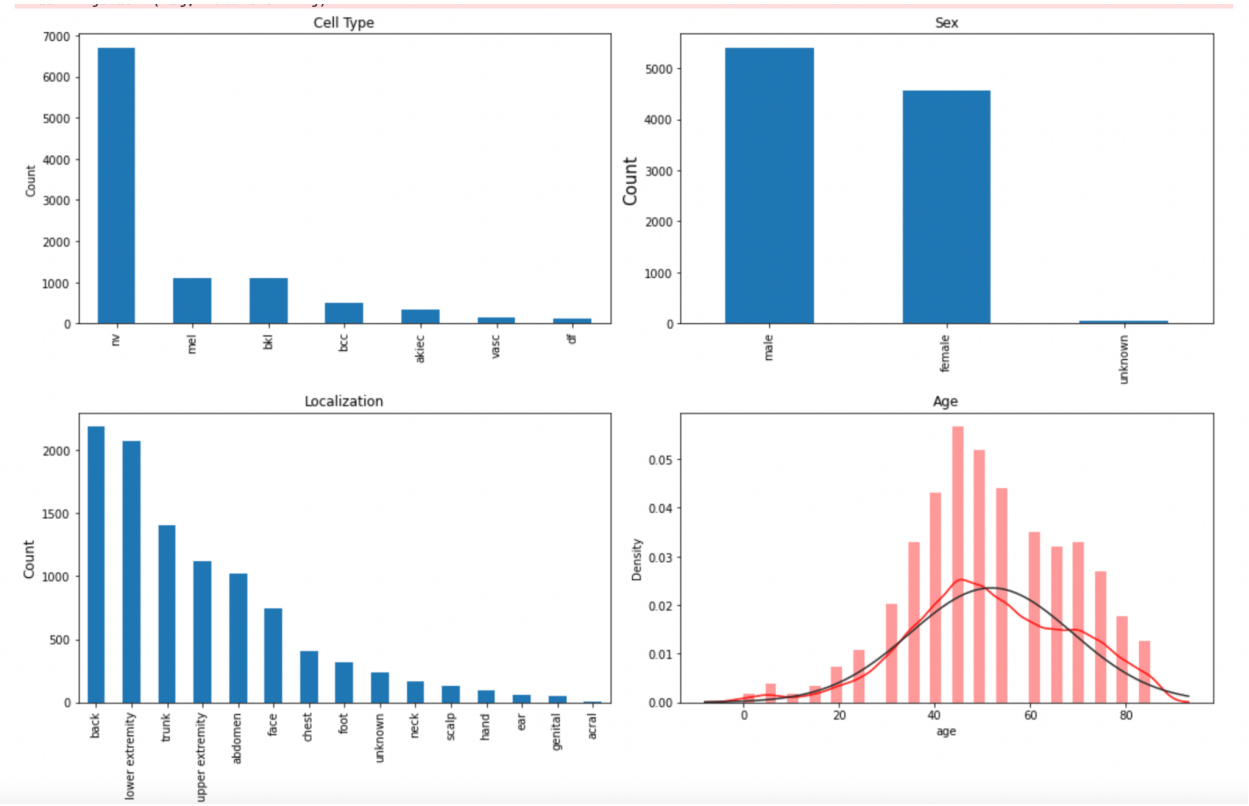
3) Preprocessing

Here we add the path of each of the image data to the right row of the meta data. By doing this we have attached each of the image data, with the right label, which is the class of the image data. And then we attach each of the image data to the corresponding row based on the image ID, and these images are stored here in the form of a numpy array. Next, we want the classes to be labeled numerically, hence we encode a numeric value for each of the labels from values 0 to 6.

(refer to the .ipynb file to view code and cell outputs)

4) Analyzing the distribution of data

Now before we actually split our dataset into test and training sets, we need to make sure that the dataset is well distributed. Because here in our analysis on the distribution of the data, we can see that the data is unbalanced.



This figure shows how unbalanced our dataset is in terms of class distribution, gender ratio, localization of skin lesions, and age group.

5) Balance the data

In this step we resample our dataset in order to get a more uniform dataset before we actually split it into a test set and training set.

(refer to the .ipynb file to view code and cell outputs)

6) Splitting of dataset into test and training sets

In the next step we are gonna convert the dataframe column of the images into a numpy array, and then split our dataset into a train set and test set with a ratio of 75 : 25.

(refer to the .ipynb file to view code and cell outputs)

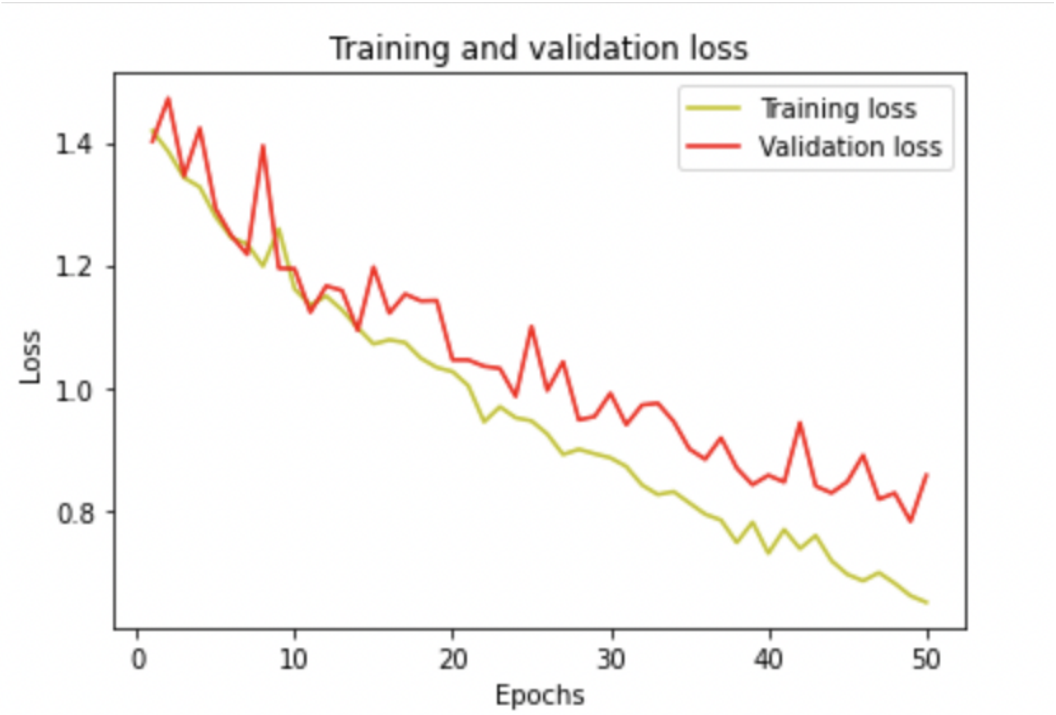
7) Define and train the model

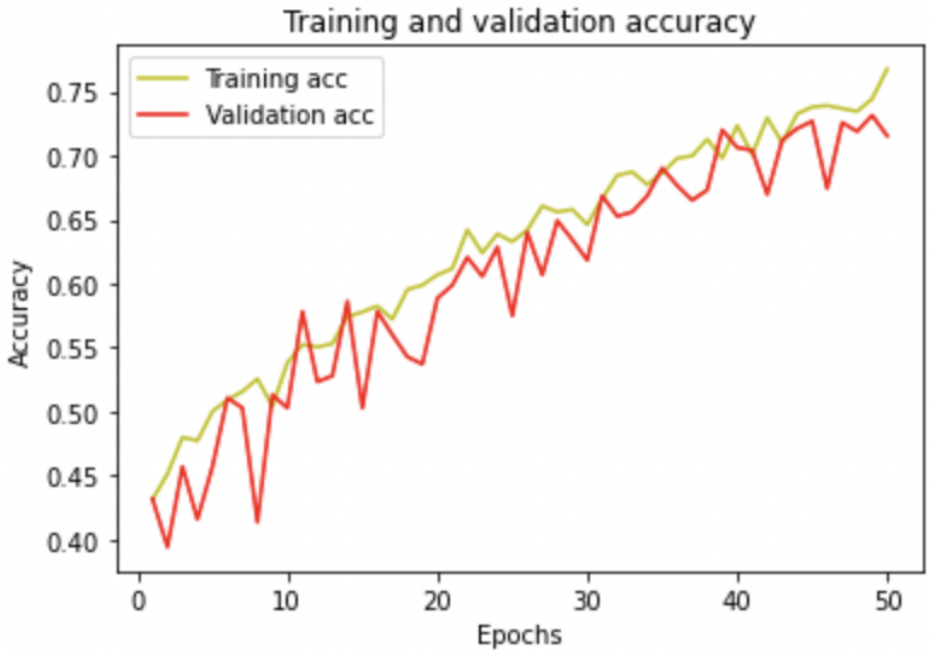
Now in the next step we are implement our sequential model for which we are using the relu activation function and Maxpool size 2 x 2. After which we train our model and set epoch as 50 which means there would be 50 iterations. After doing these 50 iterations we get the test accuracy as 0.71, which is actually really impressive.

(refer to the .ipynb file to view code and cell outputs)

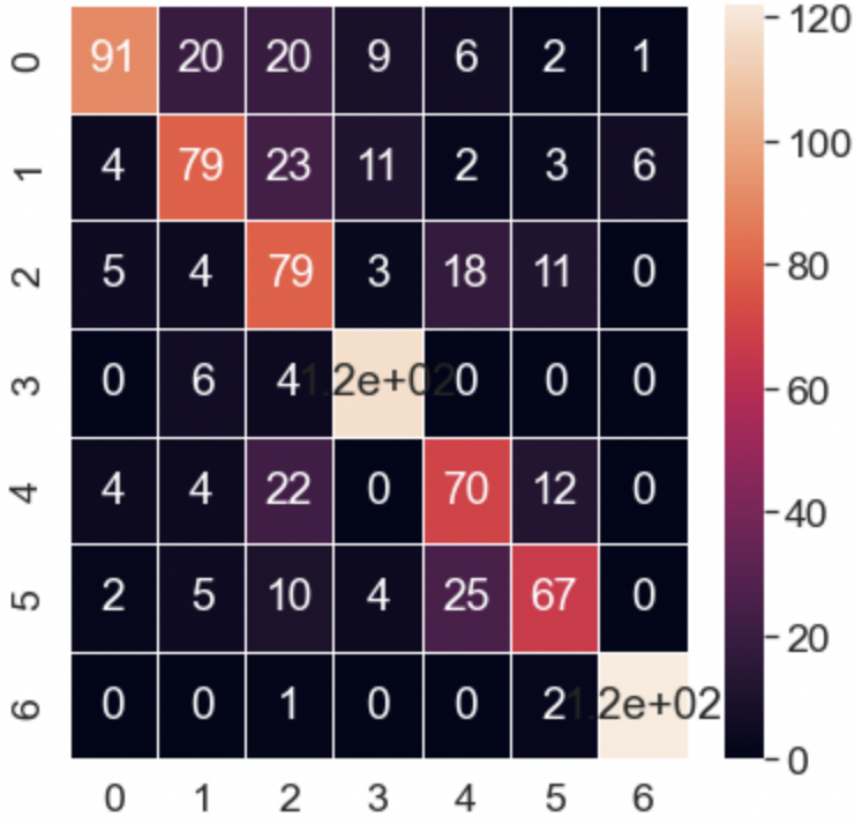
8) Model Analysis and visualization

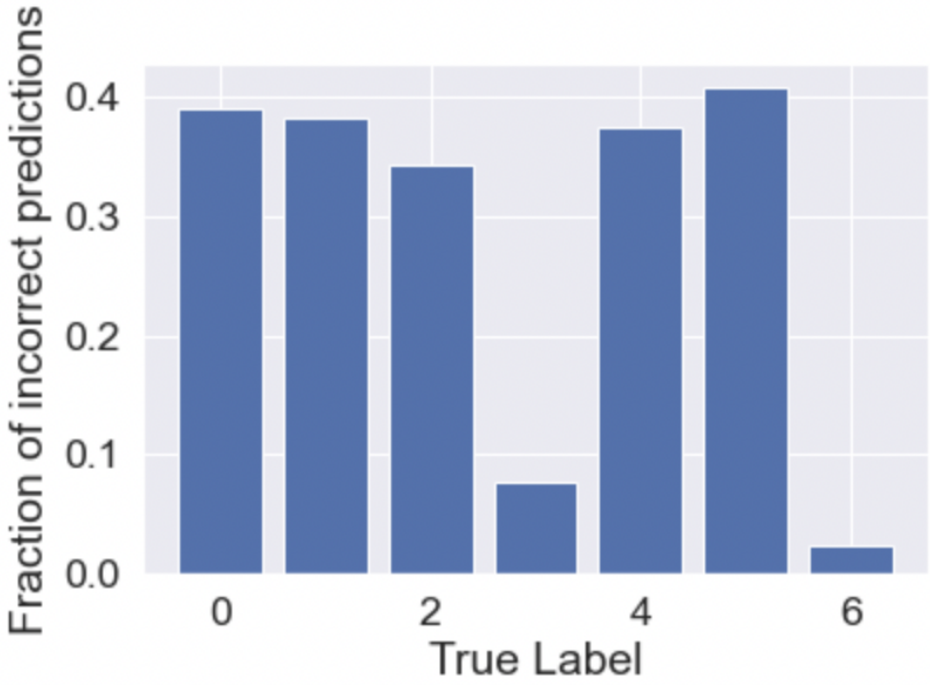
Once the model is trained on our training set and then tested on our test set, we then analyze the performance of our model by visualizing with the help of plotting graphs and confusion matrix.




Training and validation accuracy
Training acc
Validation acc
Accuracy
Epochs
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0
10
20
30
40
50

7]:


91 20 20 9 6 2 1
4 79 23 11 2 3 6
5 4 79 3 18 11 0
0 6 4 1.2e+02 0 0 0
4 4 22 0 70 12 0
2 5 10 4 25 67 0
0 0 1 0 0 2 1.2e+02
0 1 2 3 4 5 6
120
100
80
60
40
20
0



Results:

Our model performed really well with an accuracy of 0.71, which is impressive due to the fact that even the human eye finds it extremely difficult to differentiate between the different skin lesions.