Machine Learning (9860A)

Final Project Report

Skin Lesion Classification using Convolutional Neural Network Approaches

Submitted By:

Rutvik Patel

Tapan Dandawala

Submitted To:

Professor Charles Ling

Index

Abstract

Skin cancer or Cutaneous melanoma is one of the most prevalent types of cancer. However, easily treatable if detected in early stages. For rural locations or locations without adequate medical facilities, machine learning based approach, with adequate performance might help assist local medical professionals classify a skin lesion.

1. Introduction

Dermatoscopy is a popular non-invasive technique for determining the type of skin lesion. This method allows the medical practitioner to see the lesion without being obstructed by skin reflections. The accuracy of clinical diagnosis of cutaneous melanoma is highly dependent on the medical practitioner's experience and the "rareness" of the case. In this case, a machine learning model may be useful because it can help the practitioner with their diagnosis, especially if it is a rare case or they lack experience in this task.

2. Objective

- To develop Convolutional Neural Network based model to classify the type of skin lesion, using a highly imbalanced dataset of 7 different classes of dermoscopic images.
- Explore effectiveness of “weighted” approach in this case.
- Explore how the model performs if we remove the far-majority class from the given dataset.

3. Methodology

3.1 Data Source

The Data consists of more than 10,000 images from an ISIC 2018 challenge. There are two subsets (used here) of data provided with the competition, "Training" and "Validation". There are 7 classes of images which are randomly named, in the given folder.

3.2 Pre-Processing

I. The data provided consists of a folder containing dermatoscopic images and .csv files containing one-hot encoded labels for the images, for both Training and Validation sets. This information is used to separate images from different classes into 7 different folders so that the data can be used with ImageDataGenerator.

II. While importing the images with ImageDataGenerator, we would split the Training data into "Train" and "Test" with 80-20 ratio. Validation set is similarly imported but kept as it is. Further, the data is augmented to increase our samples.

III. The Images are converted to (75,100) shape, to be used as training data for the model.

Class Name	Number of Samples in Training Set
Actinic keratosis - AKIEC	262
Basal cell carcinoma - BCC	412
Benign keratosis - BKL	880
Dermatofibroma- DF	92
Melanoma - MEL	891
Melanocytic nevus - NV	5364
Vascular lesion- VASC	114

Table 1.

3.3 Model Creation and Classification

I. A CNN model is created consisting of 6 convolutional layers in total, and 2 fully connected hidden layers. Here, normalization techniques like Dropout and BatchNormalization are used.

II. Since our dataset is highly imbalanced, and considering one of our objectives, we would also consider ratios of number of samples in each class, referred to as class weights.

III. Later, the model is trained with EarlyStopping callback, twice, with and without class weights. This process is repeated for a total of 3 times, where we consider 3 variants:

   a. Minimize Loss
   b. Maximize Validation Accuracy
   c. Maximize Validation Recall

IV. Finally, we would get 6 models (a set of two for each variant)

3.4 CNN Model Breakdown

I. There are 6 convolutional layers with 16, 32, 64, 64, 128 and 128 filters respectively in each convolutional layer, 3x3 kernel size, activation by ReLU, and 4 maxpooling layer with a 2x2 pool size and stride of 1x1.

II. Conv2D() convolutional layers with input shape of (75,100, 3) with filters applied to extract different features:

III. MaxPooling2D():

   a. To reduce dimensionality of images by reducing number of pixels in output.
   b. After the convolutions, the data is passed to the dense layer to flatten and to add several Dense layers.

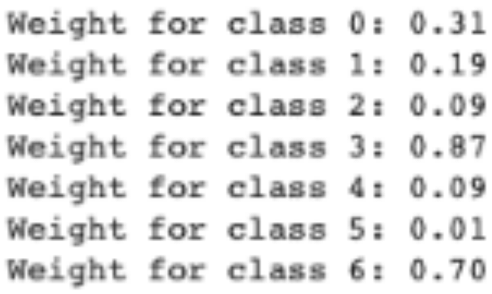
IV. Dense() layers feeds output of convolutional base to neurons:

   a. compile model using the loss function of binary_crossentropy for binary classification

V. Used SoftMax as an activation function in the last layer i.e., output layer of the CNN model

3.5 Weighted Approach

With the class distribution being highly imbalanced, class weights are used to equalize the effect of this imbalance on weight updates, while training the model. The weights used for our model is listed below:



```
Weight for class 0: 0.31
Weight for class 1: 0.19
Weight for class 2: 0.09
Weight for class 3: 0.87
Weight for class 4: 0.09
Weight for class 5: 0.01
Weight for class 6: 0.70
```

Figure 1. Weight Distribution

4. Model Evaluation and Metrics

For these models, we have considered Validation Accuracy and Validation Recall as two main metrics. Other metrics include Training and Validation Loss, Accuracy, Recall, Precision, True Positives, True Negatives, False Positives and False Negatives. To compare models, we have considered Validation set evaluation metrics, and later Test set metrics.

Here is the Validation and Test Loss, Accuracy, Recall, Precision and F1-score, for each case discussed above:

Based on the F1-score and Val Accuracy and Recall, we would consider the model where Val Recall is maximized.

		At parameters selected by EarlyStopping based on Val metrics				
Model		Val Loss	Val Accuracy	Val Recall	Val Precision	Val F1-score
Minimize Val Loss	Equal-Weighted	0.9037	0.7046	0.6839	0.7334	0.7078
	Weighted	1.77	0.3316	0.2228	0.4215	0.2915
Maximize Val Acc.	Equal-Weighted	0.6368	0.7564	0.7098	0.7740	0.7405
	Weighted	0.6954	0.7409	0.6383	0.8216	0.7184
Maximize Val Recall	Equal-Weighted	0.7160	0.7512	0.7305	0.8011	0.76424
	Weighted	0.6893	0.7461	0.7202	0.7942	0.7554

Table 2.

		At parameters selected by EarlyStopping based on Val metrics				
Model		Test Loss	Test Accuracy	Test Recall	Test Precision	Test F1-score
Minimize Val Loss	Equal-Weighted	0.7587	0.7320	0.6420	0.8295	0.7238
	Weighted	0.8770	0.6605	0.5700	0.8255	0.6744
Maximize Val Accuracy	Equal-Weighted	0.8815	0.7365	0.6965	0.7997	0.7445
	Weighted	0.6659	0.7550	0.6955	0.8236	0.7541
Maximize Val Recall	Equal-Weighted	0.8278	0.7335	0.7155	0.7727	0.7430
	Weighted	0.6745	0.7680	0.7350	0.8068	0.7692

Table 3.

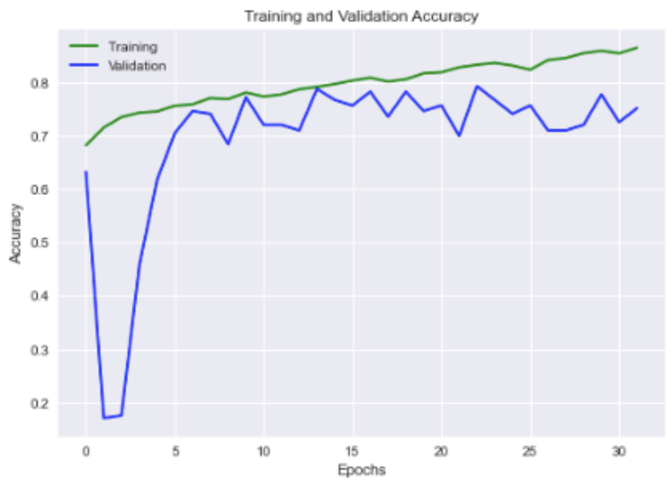


Figure 2. Training and Validation Accuracy

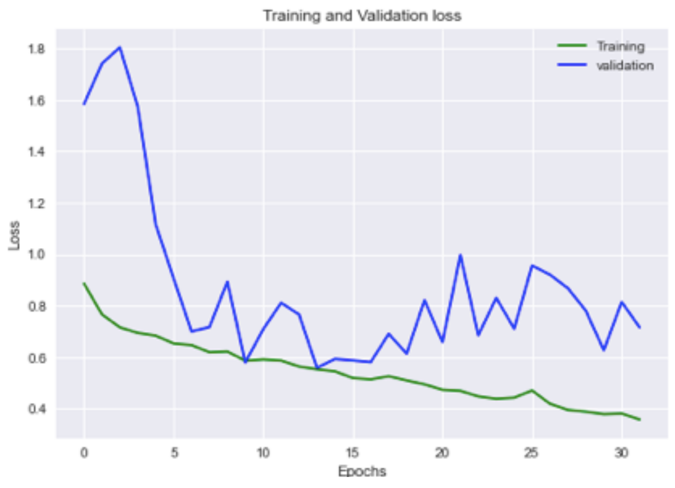


Figure 3. Training and Validation Loss

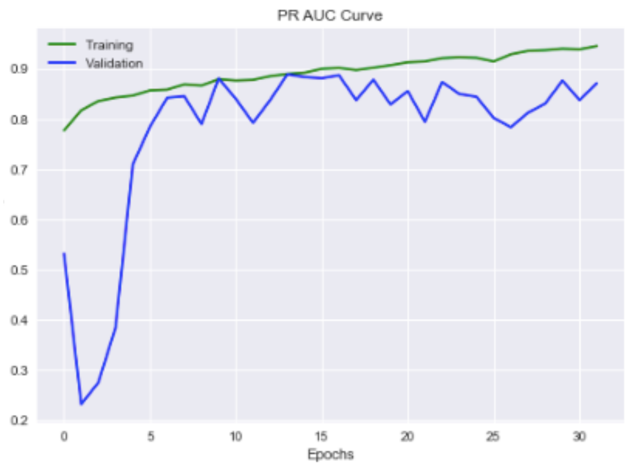


Figure 4. AUC Score for PR

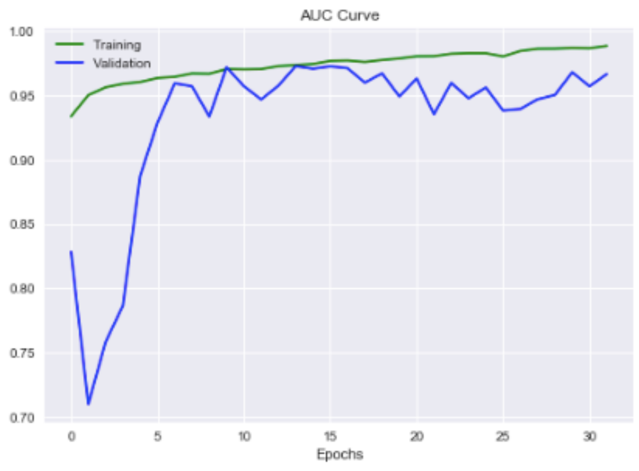


Figure 5. AUC Score

5. Different Approach

I. As, the data is imbalanced, we tried removing the far-majority class which is hugely responsible for class imbalance, NV, and tried training the model with 6 classes instead of 7. Also, we tried the same approach with transfer learning.

II. We are assuming that NV Class can be easily identified by the medical practitioner, and they might need help only to identify the other 6 types of skin lesions.

Based on the F1-score and Val Accuracy and Recall, we would consider the model where Val Recall is maximize.

		At parameters selected by EarlyStopping based on Val metrics				
Model		Val Loss	Val Accuracy	Val Recall	Val Precision	Val F1-score
Minimize Val Loss	Equal-Weighted	2.1531	0.3142	0.3142	0.3384	0.3259
	Weighted	2.002	0.3285	0.2714	0.3958	0.3220
Maximize Val Acc.	Equal-Weighted	2.6198	0.3142	0.3142	0.3188	0.3164
	Weighted	2.2999	0.4428	0.4285	0.4687	0.4476
Maximize Val Recall	Equal-Weighted	1.677	0.6142	0.6142	0.6322	0.6230
	Weighted	2.1761	0.5857	0.5571	0.5735	0.5651

Table 5.

		At parameters selected by EarlyStopping based on Val metrics				
Model		Test Loss	Test Accuracy	Test Recall	Test Precision	TestF1-score
Minimize Val Loss	Equal-Weighted	1.6895	0.3323	0.	0	
	Weighted	1.9540	0.034	0	0	
Maximize Val Accuracy	Equal-Weighted	1.7155	0.2777	0	0	
	Weighted	1.7807	0.4310	0.3778	0.4560	0.4132
Maximize Val Recall	Equal-Weighted	1.9364	0.6009	0.5948	0.6087	0.6016
	Weighted	2.23258	0.4947	0.4765	0.5024	0.4891

Table 4.

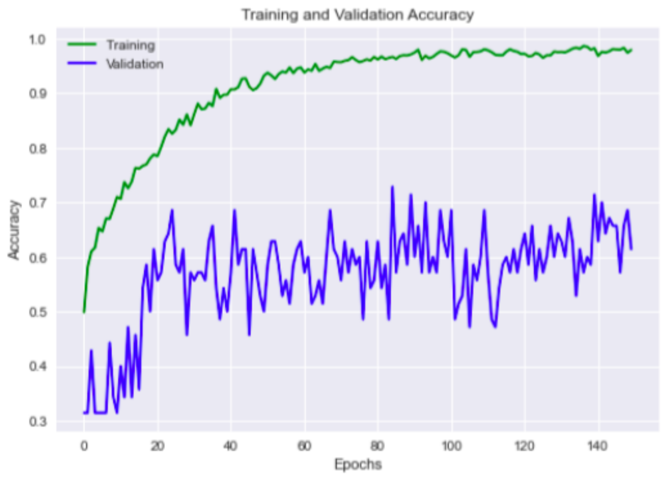


Figure 6. Training and Validation Accuracy

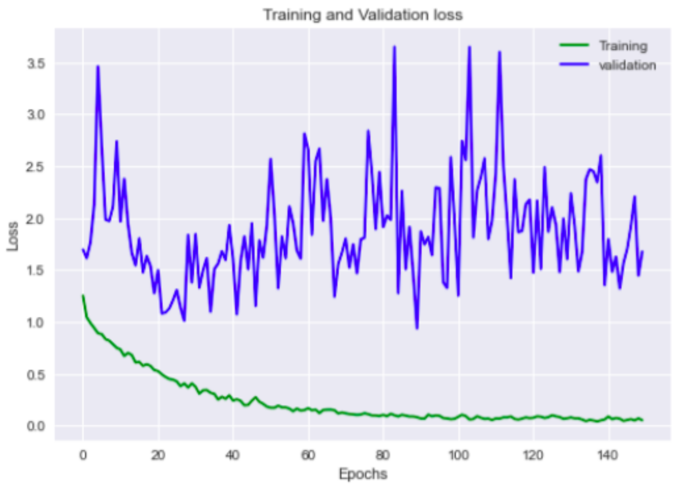


Figure 7. Training and Validation Loss

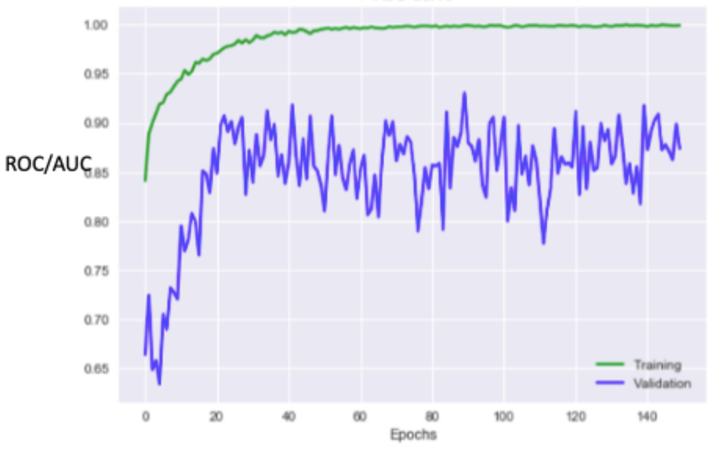


Figure 8. AUC/ROC Score

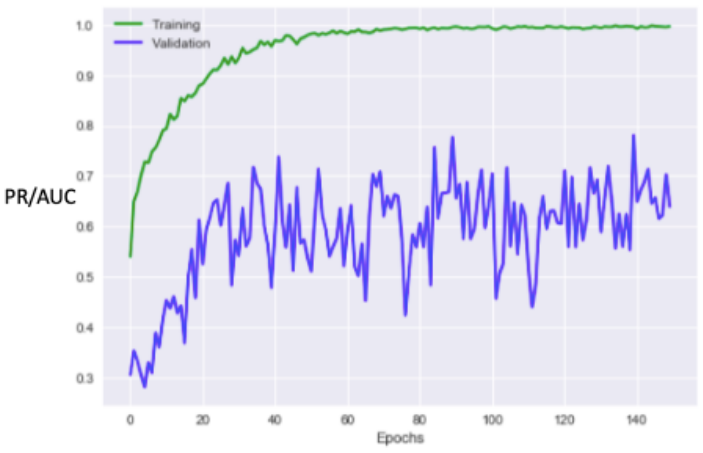


Figure 3. AUC/PR Score

6. Results

We can see that the model selected in the 4.1 produces one of the best results. The Test Accuracy for the selected model is 73.35% and Recall is 71.55%. If we consider the case where there are only 6 classes (after removing the far-majority class), we get Test Accuracy of 60.09%

It was also observed that, although transfer learning approach resulted in high validation accuracy, the model performed very poorly with test set.

7. Conclusion and Discussion

I. We can say that the most challenging part of this type of classification problem is the size and diversity of dataset. Importantly, the intrinsic property of medical images is highly irregular distribution of classes within the dataset. In our case, NV class had more samples than all other classes combined causing huge imbalance.

II. Class weights had quite small effect on our model.

III. Since this is medical data, we have tried to prioritize Accuracy and Recall.

8. Future Work

We can further improve the model by tweaking the Convolutional layers, like adding Conv layers or increasing filters. Also, we can use architectures like VGG16, VGG19 or ResNet to improve the performance for this task, especially the case where we have 6 classes, as we think the model is not able to extract enough features to make better classification. It might be possible to greatly improve results if larger training data is used.

9. References

I. *Previous personal projects related to this type of task*

II. *Python library documentation for the packages used*

III. *https://pubmed.ncbi.nlm.nih.gov/11902502/*

IV. *https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6091241/*

V. *https://challenge2018.isic-archive.com/task3/*

VI. *Other Image classification examples on Kaggle*