Skin Cancer Detection

PyTorch Capstone Project Report

Andrew Neuman

Purpose:

The goal of this project is to use existing state of the art pretrained models such as ResNet18 and Inception V3 to classify various skin lesions correctly.

Interest:

This project is of interest for several reasons. Firstly, I had a skin lesion that my family doctor wanted to have a dermatologist look at. After booking the appointment with the dermatologist, I was told the wait would be over a year. Initially, I was frustrated, but then I realized that this is something I might be able to use machine learning to solve.

The results of this research will be of value not just to me, but to anyone else who is in a similar situation and wants a rough estimate what their lesion is while waiting. It could also be used by the government, or healthcare system to try to prioritize different people based on the severity of their lesion.

Dataset:

The dataset that I am using is the ISIC 2019 challenge dataset. The dataset consists of 25,331 labeled images that are categorized into 8 distinct types of lesions. The different lesions are melanoma, melanocytic nevus, basal cell carcinoma, actinic, keratosis, benign keratosis, dermatofibroma, vascular lesion, and squamous cell carcinoma (https://challenge.isic-archive.com/landing/2019/). As per the project requirements, I will only use a maximum of 1000 images per category.

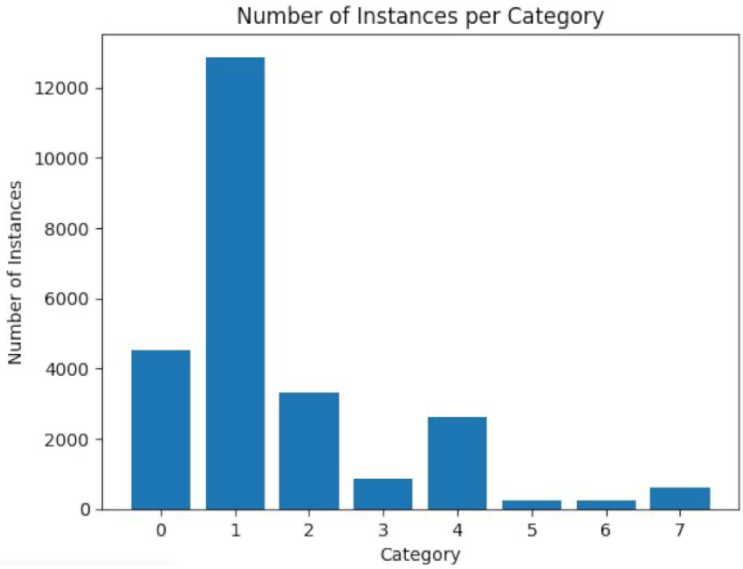
This is the full citation for the dataset:

BCN_20000 Dataset: (c) Department of Dermatology, Hospital Clínic de Barcelona

HAM10000 Dataset: (c) by ViDIR Group, Department of Dermatology, Medical University of Vienna; https://doi.org/10.1038/sdata.2018.161

MSK Dataset: (c) Anonymous; https://arxiv.org/abs/1710.05006; https://arxiv.org/abs/1902.03368

The dataset is highly imbalanced as follows:



The dataset is also a combination of different sized images ranging from 1024x1024 to 600x450. Most of the images are 1024x1024, but there are a mix of different sized images.

Data Preprocessing:

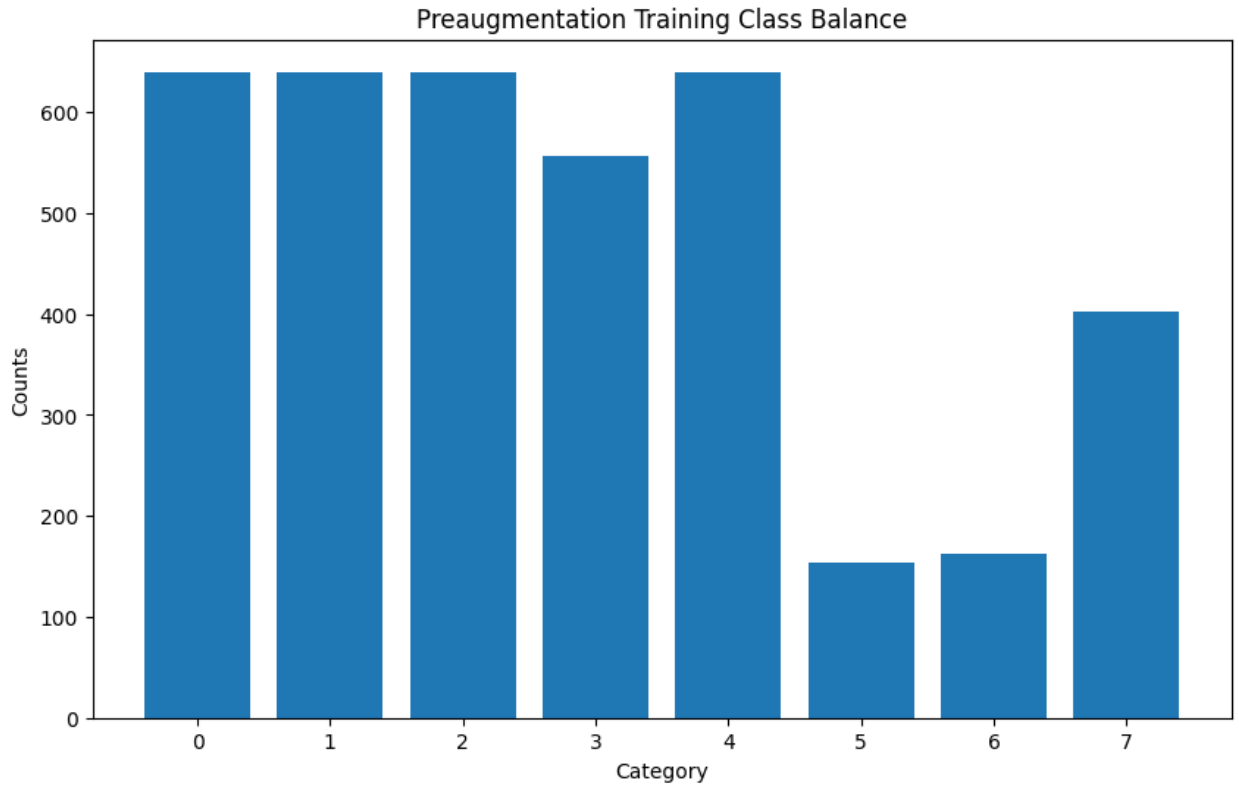
This project has gone through various preprocessing stages. Initially, I was scaling all the images down to the same size. Then images were converted to tensors, pixel values where normalized, and the dataset was split into training (64%), validation (16%), and testing (20%).

The performance of my model was not good. So, I began trying to figure out what the issue was. One source of the inferior performance was the imbalanced data set. This was addressed by using image augmentation to create altered versions of existing images in classes that were underrepresented, in the training set only. This led to a significant bump in performance, but the model was still performing in the mid 60% range.

Another source of mediocre performance was the mixed image sizes. Originally, I was just scaling all the images down and feeding them to the model. It turns out that this might have been the source of some error. The recommended approach is to either pad all the smaller images to the same size as the largest image, or to crop all the larger images down. Unfortunately, when I realized this might have been an issue there was not much time to complete the project. Since most images where 1024x1024, I decided to drop all the other images. This made my already severely imbalanced dataset even more imbalanced.

To balance the classes, I used PyTorch's built in image transformations. I tried to apply transformations that were not too intense and could be observed by a doctor in practice.

These transformations include flipping the image, rotating the image, adjusting the sharpness, adding some blur, or randomly augmenting the image.



Model Selection and Architecture:

To approach this problem, I initially planned to use state of the art pretrained models like Resnet18, and Inception V3. It was then suggested to me that I should try writing my own CNN model and comparing the results between the two.

Unfortunately, the performance of Resnet18 and Inception V3 was much poorer than I originally expected. So, I spent all my time trying to fine tune them to get better results. I did very briefly try a CNN of my own creation, but the performance was not good. It is possible that this is something I overlooked, that if I had invested more time into building my own CNN, the performance would have been better.

Model Training:

My research began when I took the transfer learning example from class and fitted it to my dataset. This involved using Resnet18 on my dataset. The loss function was cross entropy loss. The optimizer was stochastic gradient descent. The initial parameters that I used were

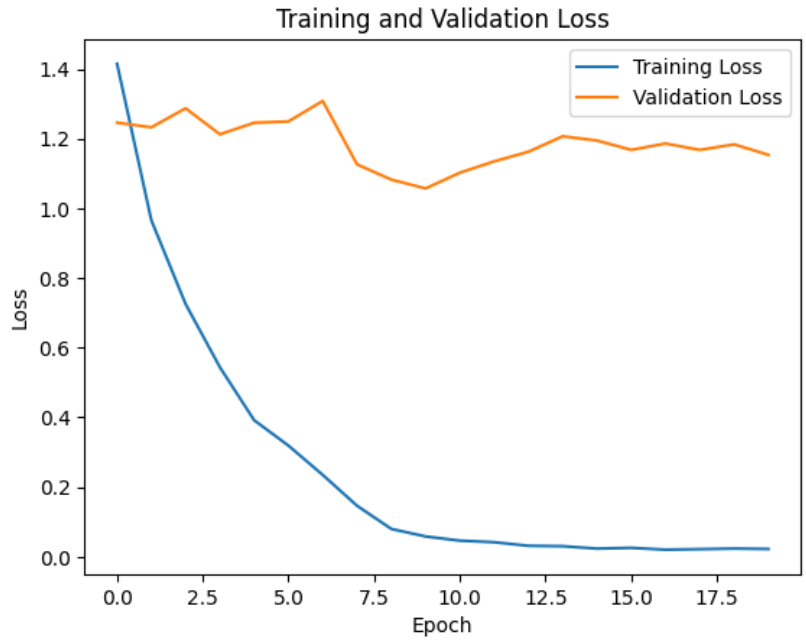
10 epochs, learning rate 0.001, batch size 32, momentum 90. The performance was an abysmal 40-50%.

This led to me trying to figure out issues with the model. Instead of only training the final layer, I tried training the entire model. This led to a performance bump somewhere around 50-55%. Fixing the imbalanced dataset, and multiple image sizes, got us around 62%. Switching from Resnet18 to InceptionV3 got us to around 68%. Fine tuning InceptionV3 got us to 70%.

Evaluation:

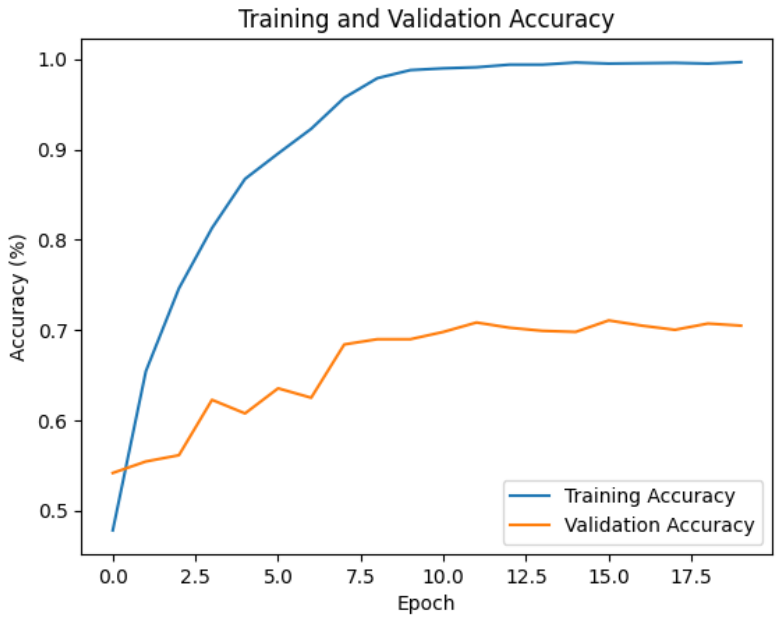
After fine tuning the model for many iterations of different batch size, learning rate, number of epoch's, weight decay, learning rate scheduling decay, the best performance I was able to get was around 70% F1 Score. The hyperparameters for this model the optimizer is ADAM, the learning rate was 0.001, no weight decay, batch size 8.

The training and validation loss functions are plotted below:



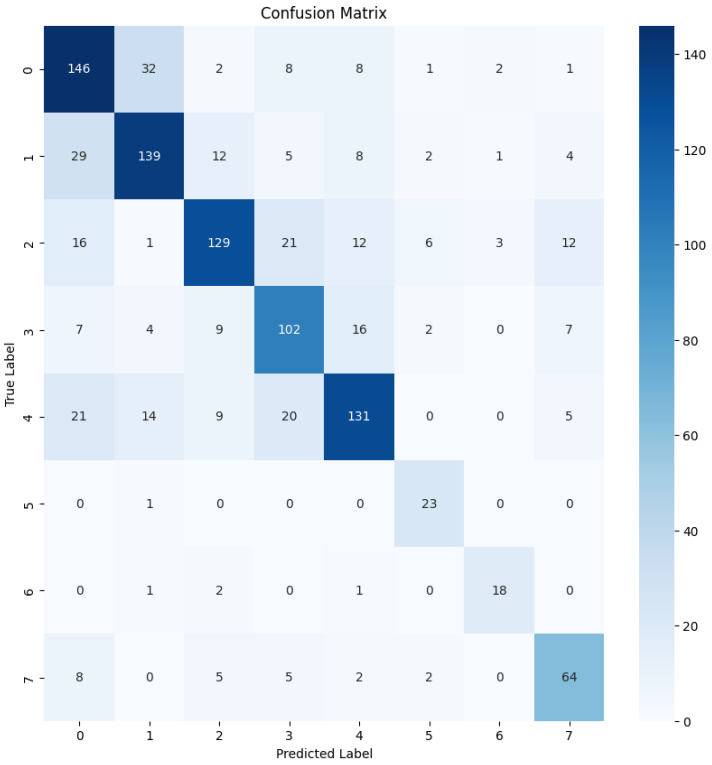
We see that the training loss is improving rapidly, while the validation loss is flat. Suggesting the existence of overfitting.

The training and validation accuracies are plotted below:



More evidence of overfitting is exhibited here.

The confusion matrix is plotted below:



Finally, the classification report is plotted:

	precision	recall	f1-score	support
0	0.64	0.73	0.68	200
1	0.72	0.69	0.71	200
2	0.77	0.65	0.70	200
3	0.63	0.69	0.66	147
4	0.74	0.66	0.69	200
5	0.64	0.96	0.77	24
6	0.75	0.82	0.78	22
7	0.69	0.74	0.72	86
accuracy			0.70	1079
macro avg	0.70	0.74	0.71	1079
weighted avg	0.70	0.70	0.70	1079

Insights Gained:

Overall, I am not impressed with the results of my project. I honestly figured that my F1 score would be in the 80-90% range. I spent a lot of time trying different models, different augmentations, mixes of image sizes, and hyper parameters. So, I am a bit disappointed in the results.

However, this is not a simple problem. The imbalanced dataset on its own is highly challenging, and the mix of image sizes only enhances that problem.

There are several things that I have learned throughout this journey. First, I may have overestimated my abilities, in the given time frame, with everything else going on. Second, dealing with complex datasets is difficult. Third, hyper parameter tuning complex models, is time consuming.

Overall, I really enjoyed this project. In its current stage it isn't capable of being used for pre-screening. Maybe this goal was too ambitious, but I am excited to continue to improve this model.

Finally, as this was a competition in 2019, there are brief papers published by the top teams. I never looked at these as I wanted to challenge myself. I am excited to see what they did and implement some of their techniques.