Melanoma Detection Assignment

Naresh Kumar Ailaveni

Problem Statement:

This assignment is to build a multiclass classification model using a custom convolutional neural network in TensorFlow.

Dataset details:

The dataset consists of 2357 images of malignant and benign oncological diseases, which were formed from the *International Skin Imaging Collaboration (ISIC)*. All images were sorted according to the classification taken with ISIC, and all subsets were divided into the same number of images, with the exception of melanomas and moles, whose images are slightly dominant.

The data set contains the following diseases:

- Actinic keratosis
- Basal cell carcinoma
- Dermatofibroma
- Melanoma
- Nevus
- Pigmented benign keratosis
- Seborrheic keratosis
- Squamous cell carcinoma
- Vascular lesion

Project Pipeline:

- Data Reading/Data Understanding→ Defining the path for train and test images

- Dataset Creation→ Create train & validation dataset from the train directory with a batch size of 32. Also, make sure you resize your images to 180*180.

- Dataset visualisation → Create a code to visualize one instance of all the nine classes present in the dataset

- Model Building & training:

    - Create a CNN model, which can accurately detect 9 classes present in the dataset. While building the model, rescale images to normalize pixel values between (0,1).
    - Choose an appropriate optimiser and loss function for model training
    - Train the model for 20 epochs
    - Write your findings after the model fit. You must check if there is any evidence of model overfit or underfit.
    - Chose an appropriate data augmentation strategy to resolve underfitting/overfitting
    -

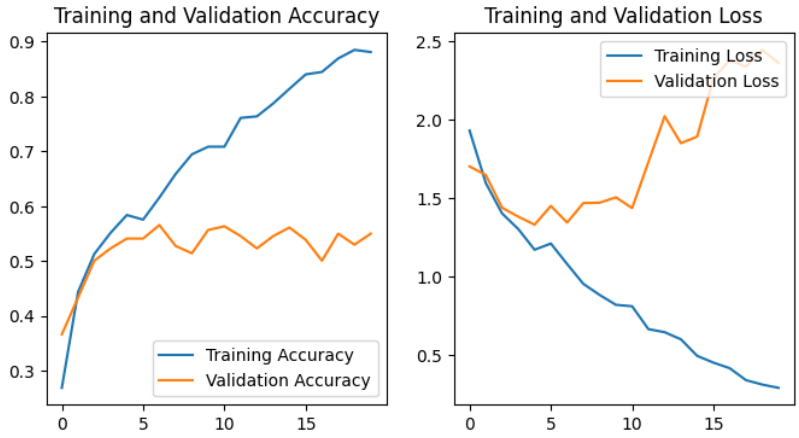
- Model Building & training on the augmented data:
    - Create a CNN model, which can accurately detect 9 classes present in the dataset. While building the model rescale images to normalize pixel values between (0,1).
    - Choose an appropriate optimiser and loss function for model training
    - Train the model for 20 epochs
    - Write your findings after the model fit, see if the earlier issue is resolved or not?
    - Class distribution: Examine the current class distribution in the training dataset
    - Which class has the least number of samples?
    - Which classes dominate the data in terms of the proportionate number of samples?
    - Handling class imbalances: Rectify class imbalances present in the training dataset with Augmentor library.
    -

- Model Building & training on the rectified class imbalance data:
    - Create a CNN model, which can accurately detect 9 classes present in the dataset. While building the model, rescale images to normalize pixel values between (0,1).
    - Choose an appropriate optimiser and loss function for model training
    - Train the model for 30 epochs
    - Write findings after the model fit and confirm if the issue is resolved.

Results: 3 different Models are created to increase the accuracy and reduce the overfitting issue

We observe gradual improvement from Model 1 to Model 3:

Model 1: Simple CNN Model - Accuracy: 0.8811 | Validation accuracy : 0.5503



Todo: Write your findings after the model fit, see if there is an evidence of model overfit or underfit

Findings:

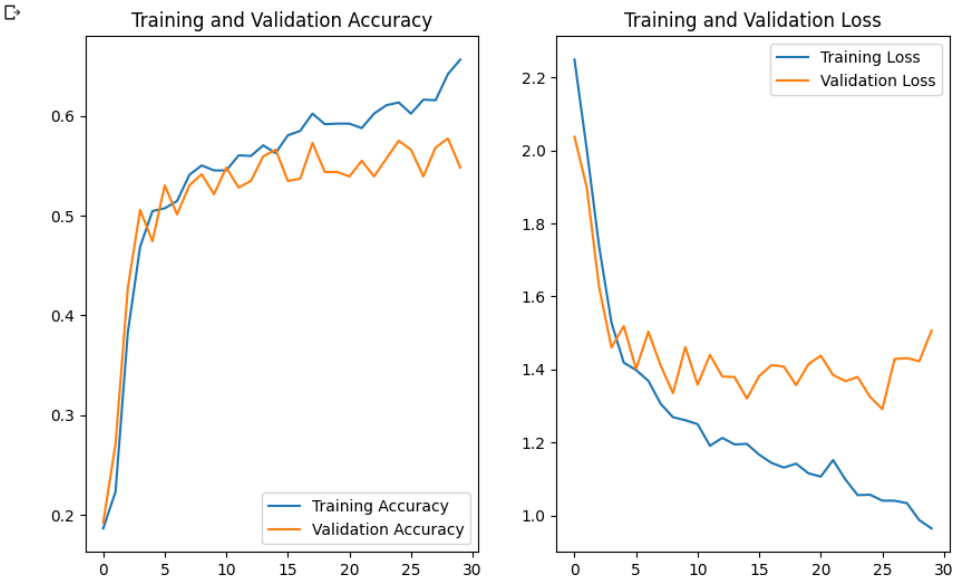
Train Data Accuracy : 0.8811

Train Loss : 0.293

Validation Data Accuracy : 0.5503

Validation Loss - 2.3615

Model 2: Augmented data with droupout - Accuracy: 0.6562 | Validation accuracy : 0.5481



Findings:

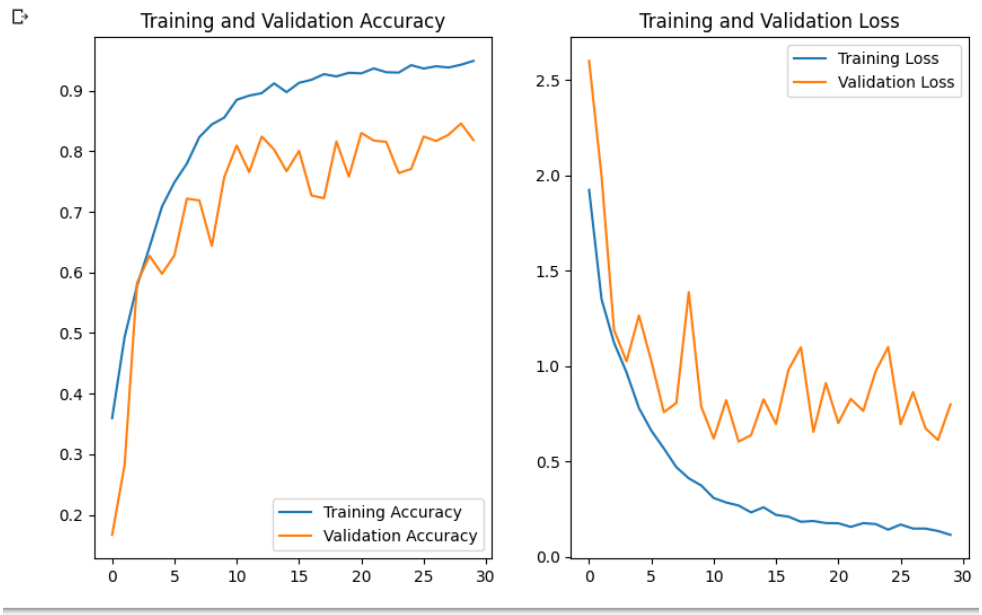
Train Data Accuracy : 0.6562

Train Loss : 0.9642

Validation Data Accuracy : 0.5481

Validation Loss - 1.5064

Model 3: Class rebalance, BatchNormalization with Dropout - Accuracy: 0.9488 | Validation accuracy : 0.8181



Findings:

Train Data Accuracy : 0.9488

Train Loss : 0.1153

Validation Data Accuracy : 0.8181

Validation Loss - 0.7992

Conclusion:

We observe gradual improvement from Model 1 to Model 3:

Model 1: Simple CNN Model

Accuracy: 0.8811 | Validation accuracy: 0.5503

Model 2: Augmented data with dropout

Accuracy: 0.6562 | Validation accuracy: 0.5481

Model 3: Class rebalance, Batch Normalization with Dropout

Accuracy: 0.9488 | Validation accuracy: 0.8181

The accuracy can still be increased using Hyper-Parameter tuning i.e., by exploring few different options like CNN configurations, loss functions, optimizers, and layer counts.