Skin Lesion Detection

Chinmay Terse
cterse@ncsu.edu
North Carolina State University
Raleigh

Eshwar Chandra Vidhyasagar Thedla
ethedla@ncsu.edu
North Carolina State University
Raleigh

Priya Sharma
psharm23@ncsu.edu
North Carolina State University
Raleigh

Rachana Sri Bollineni
rbollin@ncsu.edu
North Carolina State University
Raleigh

1 INTRODUCTION & BACKGROUND

1.1 Problem Statement

The use of deep learning in the field of image processing is increasing. Skin cancer is one of the common diseases in the community. Recent studies have demonstrated the use of convolutional neural networks (CNNs) to classify images of melanoma with accuracies comparable to those achieved by board-certified dermatologists[1]. Automated classification of skin lesions using images is a challenging task owing to the fine-grained variability in the appearance of skin lesions. Deep convolutional neural networks (CNNs) show potential for general and highly variable tasks across many fine-grained object categories. CNN takes an input image to train its network and produces an output that matches the input image using the existing image dataset. A neural network draws a path based on similarity connections of biological nerves interconnected with each other and thus creates a learning structure. The performance of CNN also varies depending on how many input images there are and how many convolutions are used [9]. Deep learning methods can be seen as a useful tool for dermatologists to detect lesions better [2].

We introduce our problem as an image classification task over the HAM10000 dataset. There are seven classes in the skin disease data set; Actinic keratoses and intraepithelial carcinoma, Basal cell carcinoma, Benign keratosis, Dermatofibroma, Melanoma, Melanocytic type and Vascular lesions. We implemented a baseline CNN on the dataset to train on and classify the images in the given dataset. Further we plan to apply strategies to handle class imbalance in our dataset using random under and over sampling, Median Frequency Balancing and data augmentation. We also plan to compare the performances on our dataset of advanced CNN models like ResNet, Xception and GoogLeNet.

1.2 Related Work

There have been many works done in the context of skin disease diagnosis using deep learning and machine learning techniques. Four different models including a classical method based on K-Nearest Neighbor with Sequential Scanning selection technique for feature selection, a classical method with complex technique KNN with Genetic Algorithm, a complex method based on Artificial Neural Networks with Genetic Algorithm to diagnose the skin cancer from the lesion images were used in [6] as opposed to the four methods we used such as CNN, ResNet, Xception and GoogLeNet. We used different advanced CNN models to check which model performed better on our dataset whereas in [7] a novel Gabor wavelet-based deep convolutional neural network is proposed for the detection of malignant melanoma and seborrheic keratosis. The proposed method is based on the decomposition of input images into seven directional sub-bands. Seven sub-band images and the input image are used as inputs to eight parallel CNNs to generate eight probabilistic predictions. In [8] they proposed a hybrid approach including faster region based convolutional neural network (RCNN), deep feature extraction, and feature selection by IcNR approach for skin lesion classification. There are many other works in close relation to our approach.

2 METHODS

Classification is an easy task for humans. Classification of images is a complex task for machines. We explain our approach and justify the rationale behind them.

2.1 Approach

2.1.1 Data Pre-Processing. The input data is the HAM10000 dataset, which is a dataset of 10,015 skin lesion images in .jpg format. Each image is a 600 x 450 px, roughly 250 KB JPEG image. In our project, we are applying the following data pre-processing methods on the skin lesion images:

- Each of the images in the training set is first normalized by subtracting the mean of the training data set, then dividing each object by the standard deviation of the set.
- In the following steps, a CNN model is built to serve as an initial baseline and then different pre-trained models are used to implement transform learning. Every image in all the sets i.e. training, validation, and test, is resized to a height of 75 px and a width of 100 px for use with the baseline CNN model, or as required by any model used in transform learning.
- The images are encoded in a (75, 100, 3) array for the baseline CNN or as required by other used models, where the last axis is used to record the color information as all our images are RGB images.

```
[[[199 142 149]  [198 143 148]  [201 144 153]  [202 145 153]  [203 145 150]
  [204 145 152]  [204 143 151]  [206 145 153]  [206 148 164]  [208 150 166]
  [212 153 165]  [211 150 164]  [214 152 170]  [215 153 168]  [211 151 166]
  [212 151 166]  [212 149 166]  [211 151 164]  [209 148 158]  [211 148 165]
  [210 148 165]  [210 149 167]  [210 148 162]  [211 149 160]  [214 154 170]
  [215 156 171]  [216 154 169]  [212 150 160]  [213 152 165]  [214 153 162]
  [212 152 162]  [214 151 161]  [214 152 164]  [217 155 168]  [216 156 164]
  [216 156 165]  [216 157 167]  [217 156 167]  [216 153 168]  [214 152 163]
  [215 154 168]  [215 153 166]  [213 152 165]  [214 153 167]  [214 154 169]
  [214 153 169]  [213 150 166]  [213 150 165]  [213 149 163]  [209 145 153]
  [210 150 160]  [212 151 162]  [212 151 164]  [210 148 157]  [208 145 156]
  [209 146 158]  [209 144 158]  [208 148 163]  [209 151 165]  [209 149 163]
  [210 150 163]  [209 150 164]  [209 147 160]  [206 148 162]  [210 150 164]
  [208 150 164]  [206 148 162]  [208 148 164]  [207 145 158]  [206 145 156]
  [205 145 157]  [207 146 161]  [206 145 163]  [204 146 161]  [203 145 157]
  [204 144 158]  [204 143 162]  [206 144 160]  [204 143 158]  [202 142 155]
  [204 144 159]  [202 141 151]  [202 143 151]  [202 142 156]  [202 142 155]
  [202 144 157]  [200 140 151]  [202 141 150]  [204 146 156]  [199 142 150]
  [200 142 154]  [198 142 156]  [195 139 155]  [194 138 152]  [197 141 151]
  [193 137 145]  [190 135 145]  [193 137 149]  [193 139 150]  [192 139
 150]]]
```

```
[[[ 0.84198702 -0.38498166 -0.23430129]  [ 0.82046125 -0.36345589 -0.25582706]  [ 0.88503855 -0.34193012 -0.14819823]
  [ 0.90656432 -0.32040436 -0.14819823]  [ 0.92809009 -0.32040436 -0.21277553]  [ 0.94961585 -0.32040436 -0.16972399]
  [ 0.94961585 -0.36345589 -0.19124976]  [ 0.99266738 -0.32040436 -0.14819823]  [ 0.99266738 -0.25582706  0.0885852 ]
  [ 1.03571892 -0.21277553  0.13163673]  [ 1.12182198 -0.14819823  0.11011097]  [ 1.10029622 -0.21277553  0.0885852 ]
  [ 1.16487352 -0.16972399  0.2177398 ]  [ 1.18639928 -0.14819823  0.17468827]  [ 1.10029622 -0.19124976  0.13163673]
  [ 1.12182198 -0.19124976  0.13163673]  [ 1.12182198 -0.23430129  0.13163673]  [ 1.10029622 -0.19124976  0.0885852 ]
  [ 1.05724468 -0.25582706 -0.0405694 ]  [ 1.10029622 -0.25582706  0.11011097]  [ 1.07877045 -0.25582706  0.11011097]
  [ 1.07877045 -0.23430129  0.1531625 ]  [ 1.07877045 -0.25582706  0.04553367]  [ 1.10029622 -0.23430129  0.00248214]
  [ 1.16487352 -0.12667246  0.2177398 ]  [ 1.18639928 -0.08362093  0.23926557]  [ 1.20792505 -0.12667246  0.19621403]
  [ 1.12182198 -0.21277553  0.00248214]  [ 1.14334775 -0.16972399  0.11011097]  [ 1.16487352 -0.14819823  0.04553367]
  [ 1.12182198 -0.16972399  0.04553367]  [ 1.16487352 -0.19124976  0.0240079 ]  [ 1.16487352 -0.16972399  0.0885852 ]
  [ 1.22945081 -0.1051467   0.17468827]  [ 1.20792505 -0.08362093  0.0885852 ]  [ 1.20792505 -0.08362093  0.11011097]
  [ 1.20792505 -0.06209516  0.1531625 ]  [ 1.22945081 -0.08362093  0.1531625 ]  [ 1.20792505 -0.14819823  0.17468827]
  [ 1.16487352 -0.16972399  0.06705944]  [ 1.18639928 -0.12667246  0.17468827]  [ 1.18639928 -0.14819823  0.13163673]
  [ 1.14334775 -0.16972399  0.11011097]  [ 1.16487352 -0.14819823  0.1531625 ]  [ 1.16487352 -0.12667246  0.19621403]
  [ 1.16487352 -0.14819823  0.19621403]  [ 1.14334775 -0.21277553  0.13163673]  [ 1.14334775 -0.21277553  0.11011097]
  [ 1.14334775 -0.23430129  0.06705944]  [ 1.05724468 -0.32040436 -0.14819823]  [ 1.07877045 -0.21277553  0.00248214]
  [ 1.12182198 -0.19124976  0.04553367]  [ 1.12182198 -0.19124976  0.0885852 ]  [ 1.07877045 -0.25582706 -0.06209516]
  [ 1.03571892 -0.32040436 -0.08362093]  [ 1.05724468 -0.29887859 -0.0405694 ]  [ 1.05724468 -0.34193012 -0.0405694 ]
  [ 1.03571892 -0.25582706  0.06705944]  [ 1.05724468 -0.19124976  0.11011097]  [ 1.05724468 -0.23430129  0.06705944]
  [ 1.07877045 -0.21277553  0.06705944]  [ 1.05724468 -0.21277553  0.0885852 ]  [ 1.05724468 -0.27735283  0.00248214]
  [ 0.99266738 -0.25582706  0.04553367]  [ 1.07877045 -0.21277553  0.0885852 ]  [ 1.03571892 -0.21277553  0.0885852 ]
  [ 0.99266738 -0.25582706  0.04553367]  [ 1.03571892 -0.25582706  0.0885852 ]  [ 1.01419315 -0.32040436 -0.0405694 ]
  [ 0.99266738 -0.32040436 -0.08362093]  [ 0.97114162 -0.32040436 -0.06209516]  [ 1.01419315 -0.29887859  0.0240079 ]
  [ 0.99266738 -0.32040436  0.06705944]  [ 0.94961585 -0.29887859  0.0240079 ]  [ 0.92809009 -0.32040436 -0.06209516]
  [ 0.94961585 -0.34193012 -0.0405694 ]  [ 0.94961585 -0.36345589  0.04553367]  [ 0.99266738 -0.34193012  0.00248214]
  [ 0.94961585 -0.36345589 -0.0405694 ]  [ 0.90656432 -0.38498166 -0.1051467 ]  [ 0.94961585 -0.34193012 -0.01904363]
  [ 0.90656432 -0.40650742 -0.19124976]  [ 0.90656432 -0.36345589 -0.19124976]  [ 0.90656432 -0.38498166 -0.08362093]
  [ 0.90656432 -0.38498166 -0.1051467 ]  [ 0.90656432 -0.34193012 -0.06209516]  [ 0.86351279 -0.42803319 -0.19124976]
  [ 0.90656432 -0.40650742 -0.21277553]  [ 0.94961585 -0.29887859 -0.08362093]  [ 0.84198702 -0.38498166 -0.21277553]
  [ 0.86351279 -0.38498166 -0.12667246]  [ 0.82046125 -0.38498166 -0.08362093]  [ 0.75588396 -0.44955896 -0.1051467 ]
  [ 0.73435819 -0.47108472 -0.16972399]  [ 0.79893549 -0.40650742 -0.19124976]  [ 0.71283242 -0.49261049 -0.32040436]
  [ 0.64825512 -0.53566202 -0.32040436]  [ 0.71283242 -0.49261049 -0.23430129]  [ 0.71283242 -0.44955896 -0.21277553]
  [ 0.69130666 -0.44955896 -0.21277553]]]
```

Figure 1: Image coordinates before and after normalization

This pre-processed data is then used as input to our baseline and other models.

2.1.2 Class Imbalance. As seen from the plot in Figure 2 of the counts of the 7 classes in the data set, we can see that there are nearly 7000 instances of lesion label nv, while the other labels are comparable to each other but highly smaller in number compared to nv. This calls for balancing the number of objects in the input that we would provide to train our classification models, as there is a possibility of learning being skewed in favor of the majority class otherwise. This phenomenon is called class imbalance, and we can implement various techniques to equalize the approximate number of objects per class that are present in the training data set.
We plan to deal with class imbalance in the following ways in our project:

- Under sampling the majority class
- Using data augmentation to over sample images of minority classes

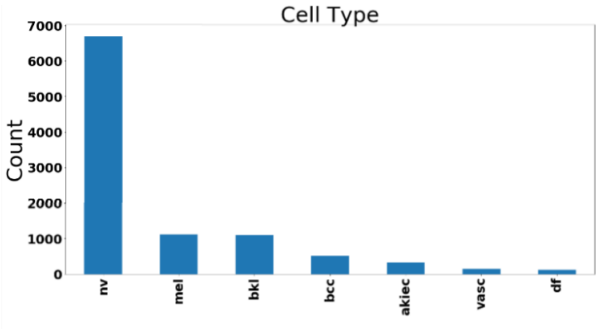


Figure 2: Count of Target Classes

2.1.3 Classification Models. Following are the steps followed as baseline before evolving to further steps.

- Baseline CNN: To serve as a baseline for class imbalance techniques performance comparison and predictions on other various pre-trained models, we built a sequential model, 4-layer convolutional neural network. We used RELU activation, and maximum pooling in the hidden layers. Model has been flattened before passing through two fully connected layers which have dense layers. We used softmax activation in the last layer, Adam optimizer to handle stochastic gradient descent method and categorical cross entropy as loss function as ours is a single-label task. We used learning rate = .001, dropout = .25, batchsize = 20 and epochs > 25. A learning rate reduction algorithm that stops the training epochs on no significant changes to evaluation metrics has also been implemented with a patience of 3.
- CNN: Popular CNN architectures for image classification like ResNet-50, GoogLeNet, Xception. Results of these CNN architectures need to be compared to evaluate the performance of the model. Convolutional neural networks have led to a series of advancements in image classification. Many other visual recognition tasks have also greatly benefited from very deep models.The training of neural networks becomes difficult, the accuracy starts saturating and also degrades due to deeper neural networks. Residual Learning tries to solve both these problems. In residual learning, instead of trying to learn features, it learns residuals. Residual can be simply understood as subtraction of features learned from input. ResNet uses shortcut connections. GoogLeNet achieves efficiency through reduction of the input image, whilst simultaneously retaining important spatial information. GoogLeNet was responsible for setting a new state-of-the-art for classification. It is the first model to come up with not only stacked up layers. Xception is a depthwise separable convolution neural network, spatial convolution performed independently on each input. It has been improved under the inception module, and residual connections.

2.1.4 Tuning Hyper-parameters. We plan to tune two hyper-parameters in our experiment:

- Batch size: Smaller batch size takes less time to train a network under the same epochs. Therefore, we will try to find

a batch size that is small yet achieves baseline performance for each model.
- Epochs: We will keep track of training and validation loss to find the right epoch count to avoid overfitting.

2.1.5 Model Evaluation.
- For all our models, we have an 80-20 split on the training and test data. Further, the training dataset is 90-10 split to create a validation dataset (10%). These processes are done in the pre-processing task and the proportions of the split are the same for all the models in the project.
- Accuracy, Precision, Recall and F1-score are all evaluated for the models under training.

2.2 Rationale

2.2.1 Data Pre-Processing.
- Normalization of the training dataset is important to even out the distribution of data in case of uneven distributions among the independent attributes, and also to scale the attribute values between predefined ranges like 0 to 1 or -1 to 1. Dataset is initially split into training and testing datasets, in a 90-10% fashion. After the split, we attempt to normalize both the training and the test datasets with the mean and the standard deviation of the training dataset only. This ensures that no future data is leaked into the training dataset, the test dataset is normalized as per our training set so the accuracy of the model is dependable, and the values are easy to work with no major information loss.
- The raw images available in the dataset are of 600 x 450 px dimensions. While the majority of the data is of the same dimensions, there are some instances of different dimensional images. Hence, we have resized our images to 75 x 100 px and designed our baseline CNN to take these dimension images as input. We chose the smallest size of the images that can be used by models so that the large dataset of around 3 gigabytes can be converted to a much smaller dataset while maintaining all original information.
- Since neural networks work with floating-point data, we have encoded all our JPEG images into (75, 100, 3) arrays as input to the classification models.
- In case of pre-processing data for pre-trained models like ResNet or GoogLeNet, images are reshaped from 600x450 px to the shape required by the particular mode. Z-score normalization and splitting proportions remain the same as applied for baseline data.

2.2.2 Class Imbalance.
- To ensure even class distribution for model training, we need to deploy some techniques to deal with the class imbalance present in our dataset. We will train our first CNN model without the application of any such technique to establish a baseline in the project so that comparisons on accuracy would be possible on the introduction of these techniques.
- A method of handling class imbalance in our application is undersampling the objects of the majority class present in the dataset. In the HAM10000 dataset, the class melanocytic nevi contains over 6500 images, while the maximum number of images in any other class is only about a thousand, as shown in Figure 2. We have undersampled this majority class so that the objects in the training dataset approximately match the number of objects in other classes in the dataset. The output baseline accuracy with sampled dataset is then compared to the accuracy of the model with imbalanced data and any other class imbalancing techniques.
- Another way of handling class imbalance implemented in the project is data augmentation, which is a process in which a variety of transformations are applied to the data to generate more data from the present dataset. Data augmentation is performed on the training set, where images are randomly flipped or shifted horizontally or vertically, zoomed, or rotated by 90 degrees randomly, thus deriving new images from the already present set of images, which in turn can be used to increase the instances for classes that have a low number of instances.

2.2.3 Classification Models. Following are the steps followed as baseline before evolving to further steps.
- After going through the CNN architecture, the Baseline CNN is built from the inspiration of AlexNet [5]. Modified the layers by decreasing a layer in convolutional and fully connected layers. We built a 4-Layer Convolutional neural network with 2-Fully connected layers. Since AlexNet implemented Rectified Linear Units (ReLU) and dropouts made us choose this as an inspiration and build a baseline model. To reduce overfitting in the fully-connected layers, AlexNet has employed a regularization method called “dropout” that proved to be very effective. The output of the last fully-connected layer is fed to a 7-way softmax which produces a distribution over the 7 class labels. To handle overfitting of the data, we use drop out = 0.25. The probability of each neuron to be neglected is 0.25 in all the layers.
- Popular CNN architectures: ResNet-50, GoogLeNet, Xception which evolved after the AlexNet. The most common architectures used for image classification will be considered for comparing the results. Transfer Learning enables us to use pre-trained models. Stacking up additional layers in the Neural Networks which results in improved accuracy and performance. The intuition behind adding more layers is that these layers progressively learn more complex features. This also results in overfitting. RestNet50 helps to skip the layer according to the values generated. GoogLeNet is used to train not only by stacking up layers but also sideways. Xception, an extreme version of inception, has the concepts of residual, depth wise separable, state of the art.

2.2.4 Tuning Hyper-parameters. Tuning batch size and epoch will give an estimate on how to train a model in less time and achieve good performance. Epoch value has been set to 25 to understand the plots correctly. For the epochs greater than 30, the learning rate is very low. For the epochs less than 25, the plots are discrete. As training CNNs on image dataset requires high computing power, we chose commonly used values for other hyperparameters.

2.2.5 Model Evaluation. Since we will ensure equal class distribution in our training data, accuracy, which is defined as the ratio of

correct predictions to all predictions, can be used to evaluate models in our project. Also since we are establishing one baseline on a highly imbalanced dataset, other evaluation metrics like precision, recall and F1-score are also implemented to be calculated by the model for every epoch.

3 EXPERIMENT

Here we have included the explanation of our dataset, the hypotheses we will use, and the design specifications we have based our experiment upon.

3.1 Dataset

We are using the HAM10000 dataset (Humans Against Machines 10000) for this project. This is taken from Harvard Dataverse (https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/DBW86T&version=3.0). This dataset is a huge collection of multi-source dermatoscopic images of common pigmented skin lesions. It contains a total of 10015 dermatoscopic images which were made publicly available by the Harvard database in June 2018. It also contains a metadata file with demographic information of each lesion. As far as the collection of dataset is concerned, in this dataset, more than 50% of lesions are confirmed through histopathology (histo), the rest of the cases is either follow-up examination (follow_up), expert consensus (consensus), or confirmation by in-vivo confocal microscopy (confocal). Using the metadata, we have also performed Exploratory Data Analysis across different attributes. It is important to understand the information in the metadata to decide which parts of the metadata we can use as a feature for our learning process. Based on our EDA, we receive following results : Basic Data Analysis:

```
Number of Samples: 10015,
Number of Features: 10,
Duplicated Entries: 0,
Null Entries: 57,
Number of Rows with Null Entries: 57 0.6%
```

Figure 3: Basic EDA

	Name	dtypes	Missing	Uniques
0	lesion_id	object	0	7470
1	image_id	object	0	10015
2	dx	object	0	7
3	dx_type	object	0	4
4	age	float64	57	18
5	sex	object	0	3
6	localization	object	0	15
7	dataset	object	0	4

Figure 4: Metadata Attributes Summary Table

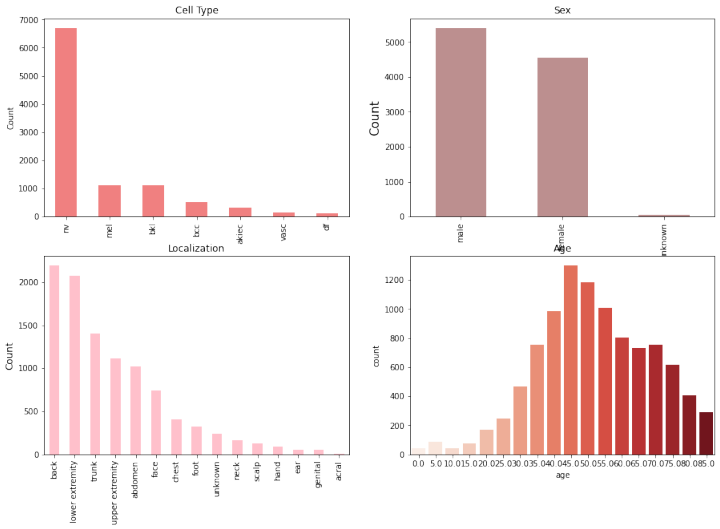


Figure 5: Metadata Attributes Counts

3.2 Hypotheses

We have tried to investigate the following hypotheses in our experimentation in the project:

- Confirm that working with a highly imbalanced dataset causes issues in training a deep learning model, affecting the ability of the model to classify objects accurately due to the training being skewed in the favour of the majority class.
- Confirm that techniques to deal with class imbalance can improve the accuracy to a significant rate, and data augmentation would be more effective than undersampling the majority classes for highly imbalanced datasets.
- For baseline CNN, the current epoch is good but a precise epoch for the data can further improve the outcomes of pre-trained models.

- Implementing transform learning by using pre-trained models like ResNET, XCeption and GoogLeNet should give a better performance than our baseline CNN. These pre-trained models will give better outcomes than the baseline approach.

3.3 Experimental Design

We have a dataset containing 10,015 raw JPEG images in over 3 gigabytes in size, along with a metadata file that serves as the specification of ground truth for each image in the dataset. Since this is a huge amount of data, pre-processing and training the data takes a large amount of time. Estimated data-preprocessing times for applying the above steps was around 10 minutes with around 12.72 Gigabytes of RAM. We are using 80% of the dataset as a training set, and the remaining 20% is maintained as the training dataset. Of the total training data, 10% of the data is again kept aside as a validation set.

.

Baseline CNN:

We built a sequential convolutional neural network model to train and test the data. Since the sequential model is useful for categorical image classification with one input and one output per neuron. In neural network design, its architecture is a series of Convolutional Layer, Batch normalization to handle the weight imbalance and Relu Layer followed by maximum pooling layer whose padding is valid. Four layers of conv2D, batch normalization, ReLU, max pooling 2D, and drop out (which is 0.25) is followed by a layer that flattens the data. The flattened data goes through 2 fully connected layers with 256, 512 nodes. The fully connected layers are followed by batch normalisation, activation, dropout. The output layer is a fully connected layer with softmax activation function, classified into 7 classes.

.

Improving CNN performance:

According to ImageNet Large Scale Visual Recognition Challenge[10], VGGNet, ResNet, GoogLeNet and Xception CNNs perform better in image classification than baseline CNNs. This is further evaluated in section 5.1.2. Improve on the baseline model with better pre-trained models

.

Class Imbalance Handling:

See section 5.1.1. Handling Class Imbalance and Data Augmentation

.

Reducing Epochs:

Learning rate reduction and early stopping callback functions are used to reduce the number of epochs whenever possible.

4 RESULTS

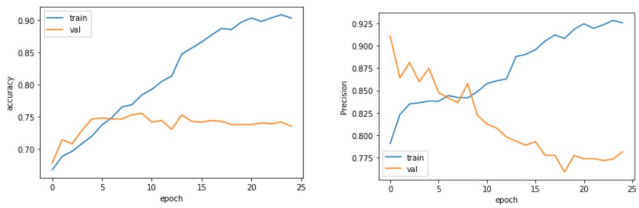
Figure 2 shows the original unbalanced dataset.

4.1 Baseline CNN trained with original dataset

Following are the values and plots of accuracy, precision, recall and F1-score when the original unbalanced dataset is used to train the baseline CNN.

```
Accuracy: 0.903190016746521
loss: 0.24585755169391632
f1_m: 0.904407262802124
precision_m: 0.9252865314483643
recall_m: 0.8864074945449829
```

Figure 6: Baseline Train Evaluation Statistics for Unbalanced Dataset



(a) Train and Validation Accuracy **(b) Train and Validation Precision**

(c) Train and Validation Recall **(d) Train and Validation F1-score**

Figure 7: Baseline Metrics Evaluation Plots with Unbalanced Dataset

```
loss: 0.9984904527664185
accuracy: 0.745881199836731
f1_m: 0.7476173639297485
precision_m: 0.772609293460846
recall_m: 0.7249895930290222
```

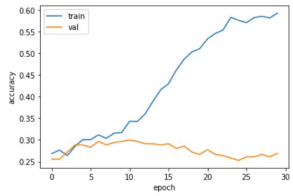
Figure 8: Baseline Test Evaluation Statistics for Unbalanced Dataset

4.2 Baseline CNN trained with under-sampled dataset

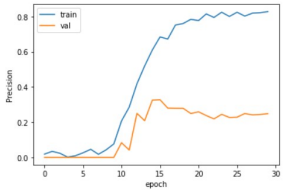
Following are the values and plots of accuracy, precision, recall and F1-score when the under-sampled majority class is used to train the baseline CNN.

```
Accuracy: 0.5926605463027954
loss: 1.0505861043930054
f1_m: 0.5185707211494446
precision_m: 0.8272427916526794
recall_m: 0.3908257782459259
```

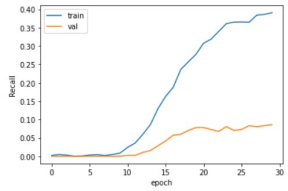
Figure 9: Baseline Train Evaluation Statistics with Undersampled Data



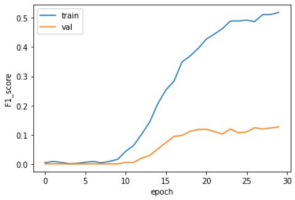
(a) Train and Validation Accuracy



(b) Train and Validation Precision



(c) Train and Validation Recall



(d) Train and Validation F1-score

Figure 10: Baseline Metrics Evaluation Plots with Undersampled Data

```
loss: 2.4477617740631104
accuracy: 0.19620569050312042
f1_m: 0.08940988779067993
precision_m: 0.2573922276496887
recall_m: 0.055894944816827774
```

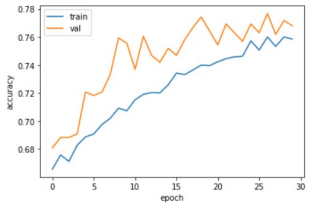
Figure 11: Baseline Test Evaluation Statistics for Undersampled Data

4.3 Baseline CNN trained with augmented dataset

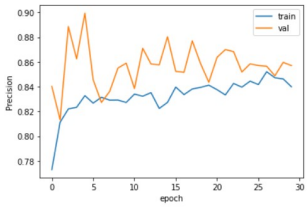
Following are the values and plots of accuracy, precision, recall and F1-score when the data augmented dataset is used to train the baseline CNN.

```
Accuracy: 0.7585298418998718
loss: 0.646212100982666
f1_m: 0.746654748916626
precision_m: 0.8398568034172058
recall_m: 0.6797504425048828
```

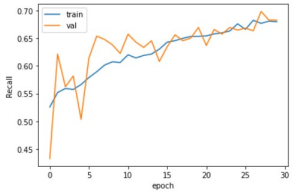
Figure 12: Baseline Train Evaluation Statistics with Augmented Data



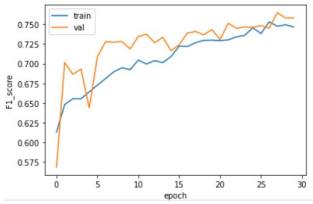
(a) Train and Validation Accuracy



(b) Train and Validation Precision



(c) Train and Validation Recall



(d) Train and Validation F1-score

Figure 13: Baseline Metrics Evaluation Plots with Augmented Data

```
loss: 0.682296633720398
accuracy: 0.7558662295341492
f1_m: 0.7449122667312622
precision_m: 0.8418228030204773
recall_m: 0.6709221005439758
```

Figure 14: Baseline Test Evaluation Statistics for Augmented Data

4.4 GoogLeNet

Following are the values and plots of accuracy, precision, recall and F1-score when the data augmented dataset is used to train GoogLeNet.

```
Accuracy: 0.7482662796974182
loss: 0.6977332234382629
f1_m: 0.737145185470581
precision_m: 0.8304170966148376
recall_m: 0.6696264743804932
```

Figure 15: GoogLeNet Train Evaluation Statistics with Augmented Data

(a) Train and Validation Accuracy **(b) Train and Validation Precision**

(c) Train and Validation Recall **(d) Train and Validation F1-score**

Figure 16: GoogLeNet Metrics Evaluation Plots with Augmented Data

```
loss: 2.0555758476257324
main_loss: 0.7274333834648132
aux1_loss: 0.6599315404891968
aux2_loss: 0.668210506439209
main_accuracy: 0.7448827028274536
main_f1_m: 0.7495604753494263
main_precision_m: 0.833039402961731
main_recall_m: 0.6838189363479614
aux1_accuracy: 0.7718422412872314
aux1_f1_m: 0.7642954587936401
aux1_precision_m: 0.8693410158157349
aux1_recall_m: 0.6856464147567749
aux2_accuracy: 0.7708437442779541
aux2_f1_m: 0.759667694568634
aux2_precision_m: 0.8628720641136169
aux2_recall_m: 0.6811821460723877
```

Figure 17: GoogLeNet Test Evaluation Statistics for Augmented Data

4.5 ResNet-50

Following are the values and plots of accuracy, precision, recall and F1-score when the data augmented dataset is used to train ResNet-50.

```
dict_keys(['loss', 'accuracy', 'f1_m', 'precision_m', 'recall_m', 'val_loss', 'val_accuracy', 'val_f1_m', 'val_precision_m', 'val
_recall_m', 'lr'])
Accuracy: 0.747295392433167
loss: 0.689801812171936
f1_m: 0.7354151606559753
precision_m: 0.8354945182800293
recall_m: 0.6633846759796143
```

Figure 18: ResNet-50 Train Evaluation Statistics with Augmented Data

(a) Train and Validation Accuracy **(b) Train and Validation Precision**

(c) Train and Validation Recall **(d) Train and Validation F1-score**

Figure 19: ResNet-50 Metrics Evaluation Plots with Augmented Data

```
loss: 0.6498482823371887
accuracy: 0.7693459987640381
f1_m: 0.7572252750396729
precision_m: 0.8591374754905701
recall_m: 0.6795374155044556
```

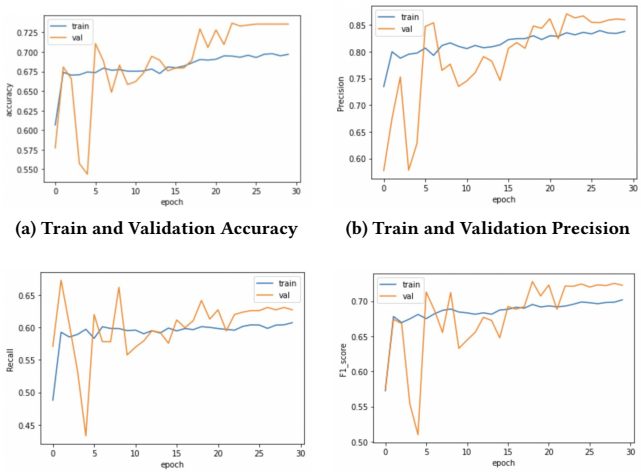
Figure 20: ResNet-50 Test Evaluation Statistics for Augmented Data

4.6 Xception

Following are the values and plots of accuracy, precision, recall and F1-score when the data augmented dataset is used to train Xception.

```
dict_keys(['loss', 'accuracy', 'f1_m', 'precision_m', 'recall_m', 'val_loss', '
Accuracy: 0.6969487071037292
loss: 0.8401266932487488
f1_m: 0.7018252015113831
precision_m: 0.8378049731254578
recall_m: 0.6074913740158081
```

Figure 21: Xception Train Evluation Statistics with Augmented Data



(a) Train and Validation Accuracy

(b) Train and Validation Precision

(c) Train and Validation Recall

(d) Train and Validation F1-score

Figure 22: Xception Metrics Evaluation Plots with Augmented Data

```
loss: 2.837921142578125
accuracy: 0.7034448385238647
f1_m: 0.7073573470115662
precision_m: 0.8478389978408813
recall_m: 0.6102234721183777
```

Figure 23: Xception Test Evaluation Statistics for Augmented Data

Table 1: Test Data Results

Model	Accuracy	F1 score	Precision	Recall
Baseline CNN	0.756	0.745	0.842	0.671
GoogLeNet	0.746	0.750	0.833	0.603
Xception	0.703	0.707	0.848	0.610
ResNet-50	0.769	0.757	0.859	0.679

5 DISCUSSION

First, we used our original dataset which has high class imbalance to train our baseline CNN. From the given plots of accuracy, precision, recall and F1-score, we saw that even though the accuracy for the training data increased, the model performs very poorly on validation data. This can be due to the model trying to learn the values in the training dataset and thus overfitting to the majority class in the dataset. Also logically, when one class is present in a huge majority, a simple function that always returns true would also have a huge accuracy, without even having to learn any data characteristics. We can see a similar situation when we try to alleviate the class imbalance by undersampling the majority class. When we undersample the majority class, we remove objects from the class having a large number of objects to match the count of objects in small classes. This however, may lead to removal of a large or even majority of our training data, or removal of important and unique data that would prove essential for the model to better predict data in the future. On the other hand, we can see that using data augmentation to increase the number of minority classes in the dataset gives a better overall performance across all metrics viz. Accuracy, precision, recall and F1-score. Even though the accuracy seems lesser than when the model was trained with unbalanced data, the validation and training metric curves both show a steady rise, as opposed to the training with unbalanced data. A fair balance is maintained in the dataset among all classes, and there is no data loss as in the case of undersampling. The dataset was unbalanced, with the Melanocytic Nevi being the majority of the samples. For such cases, the accuracy metric can give us a false perception of the model reliability. Thus, we made use of other metric scores (F1-Score, Precision, Recall) to get a better idea of our model behaviour. Although the results vary quite randomly, ResNet50 performs better than the baseline CNN model presented earlier. The test results vary within the range of 76-77% accuracy, on a higher edge as compared to the baseline CNN wherein we received a test accuracy of 75% after performing data augmentation. The dataset was imbalanced with the Melanocytic Nevi. Since it is imbalanced, the accuracy metric can give us a false perception of the model reliability. Other metric scores (F1-Score, Precision, Recall) can provide a better idea of our model behaviour. The F1 score takes into account precision and recall metrics, being more appropriate for unbalanced datasets. When we consider F1-score, the ResNet50 has the best score when compared to other models. Even though there is a high difference between train and test metrics, this model does not show the overfitting issues.

6 CONCLUSION

From running the original unbalanced dataset on the baseline CNN and comparing undersampling and data augmentation as methods with class imbalance, we understand that having a highly imbalanced dataset negatively affects the learning of the model, which may lead to overfitting or generalization. We also saw that it is not wise to undersample the majority class and remove important information that may help our model to learn important features, but use oversampling or data augmentation techniques to match the count of the minority classes in the dataset to the majority classes so that learning is performed fairly and without bias.

The Baseline CNN has shown considerate outputs. Along with the baseline technique, we have trained on popular models. Single model accuracy seemed to achieve a plateau after some epochs. The models have shown different outcomes. The training, validation and test metrics differ by very little value. Usage of the ResNet50 pretrained model has shown improvements.

7 MEETING SCHEDULE

All meetings were conducted over Zoom and were attended by all team members during the following timings:

Table 2: Meeting Schedule

Date	Agenda	Duration	Team Members
04-09-2021	Handling Class Imbalance and comparing results on baseline model	1pm-5pm	Chinmay, Eshwar, Priya, Rachana
04-14-2021	Training and evaluating ResNet performance	1pm-5pm	Chinmay, Eshwar, Priya, Rachana
04-16-2021	Training and evaluating GoogleNet performance	1pm-5pm	Chinmay, Eshwar, Priya, Rachana
04-17-2021	Training and evaluating Xception performance	1pm-5pm	Chinmay, Eshwar, Priya, Rachana
04-25-2021	Discussing Results	1pm-5pm	Chinmay, Eshwar, Priya, Rachana
04-29-2021	Writing Report	1pm-5pm	Chinmay, Eshwar, Priya, Rachana

8 GITHUB LINK

The project code is present in the following NCSU GitHub repository: https://github.ncsu.edu/cterse/alda-project.git

REFERENCES

[1] Titus J. Brinker, Achim Hekler, Alexander H. Enk, Joachim Klode, Axel Hauschild, Carola Berking, Bastian Schilling, Sebastian Haferkamp, Dirk Schadendorf, Stefan Fro hling, Jochen S. Utikal , Christof von Kalle , Collaborators, A convolutional neural network trained with dermoscopic images performed on par with 145 dermatologists in a clinical melanoma image classification task
https://doi.org/10.1016/j.ejca.2019.02.005
.

[2] Kemal Polat, Kaan Onur Koc (2020). Detection of Skin Diseases from Dermoscopy Image Using the combination of Convolutional Neural Network and One-versus-All. Journal of Artificial Intelligence and Systems, 2, 80–97.
.

[3] V. Badrinarayanan, A. Kendall and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 12, pp. 2481-2495, 1 Dec. 2017
doi: 10.1109/TPAMI.2016.2644615.
.

[4] Mateusz Buda, Atsuto Maki, Maciej A. Mazurowski, A systematic study of the class imbalance problem in convolutional neural networks, Neural Networks, Volume 106, 2018, Pages 249-259, ISSN 0893-6080
https://doi.org/10.1016/j.neunet.2018.07.011
.

[5] Krizhevsky, Alex Sutskever, Ilya Hinton, Geoffrey. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Neural Information Processing Systems. 25. 10.1145/3065386.
.

[6] Suhail M. Odeh, Abdel Karim Mohamed Baareh, A comparison of classification methods as diagnostic system: A case study on skin lesions, Computer Methods and Programs in Biomedicine, 137, 2016, 311-319
https://doi.org/10.1016/j.cmpb.2016.09.012
.

[7] Sertan Serte, Hasan Demirel, Gabor wavelet-based deep learning for skin lesion classification, Computers in Biology and Medicine, 113, 2019, 103423
https://doi.org/10.1016/j.compbiomed.2019.103423

[8] Muhammad Attique Khan, Muhammad Sharif, Tallha Akram, Syed Ahmad Chan Bukhari, Ramesh Sunder Nayak, Developed Newton-Raphson based deep features selection framework for skin lesion recognition, Pattern Recognition Letters, 129, 2020, 293-303.
https://doi.org/10.1016/j.patrec.2019.11.034
.

[9] Convolutional neural networks in dermatology
https://dermnetnz.org/topics/convolutional-neural-networks-in-dermatology/
.

[10] A compiled visualisation of the common convolutional neural networks
https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614de971