Automated Skin Cancer Detection

Aditya Prasad
(2022036)
Upal Majumder
(2021214)

1. Problem Statement:

Skin cancer, particularly malignant melanoma, poses a significant health threat with increasing instances reported annually. Detecting skin lesions, a precursor to skin cancer, is challenging due to artifacts, low contrast, and visual similarities with benign features like moles or scars. Manual inspection lacks precision, often leading to misdiagnosis and adverse outcomes, including fatalities. The imperative for accurate and efficient detection is evident, as early identification greatly improves survival rates. However, existing methods face hurdles in effectively differentiating between malignant and benign lesions, necessitating automated approaches for enhanced reliability.

2.Literature Review:

Most of the research in skin cancer detection and classification is done on applying Deep Learning algorithms or applying Deep learning model results on classical Machine learning models. [1] proposes a deep learning model which uses multiple CNN architectures like AlexNet, VGGNet etc. to train their model and come up with accuracy of 92.70%. [2] proposes a MobileNet(CNN architecture) model in which after edge based filtering, an accuracy of 88.70% was achieved and on data augmentation, an accuracy of 97.58% was achieved. [3] proposes an ANN based network which achieves an accuracy of 95.43%.
Apart from that, many papers([4], [5], [6]) have been developed which mainly focus on Deep learning models.

Our aim in this project is to come up with methods, which emphasizes on classical ML based models and assess how they perform as compared to the DL model

3. Dataset:

The dataset used for this project is the HAM10000 dataset. There are a total of 10015 data samples in this data. The dataset consists of two parts.:-

1) **HAM10000_metadata.csv**: It consists of tabular data like patient id, class id, sex, place of cancer etc. and the class dx which signifies the skin cancer disease the patient is suffering from, with outliers being a part of this dataset. A total of 6 features have been assessed and encoded(in our project) for the analysis of the data. Below image gives a rough idea of the dataset:-

	lesion_id	image_id	dx	dx_type	age	sex	localization	dataset
0	HAM_0000118	ISIC_0027419	bkl	histo	80.0	male	scalp	vidir_modern
1	HAM_0000118	ISIC_0025030	bkl	histo	80.0	male	scalp	vidir_modern
2	HAM_0002730	ISIC_0026769	bkl	histo	80.0	male	scalp	vidir_modern
3	HAM_0002730	ISIC_0025661	bkl	histo	80.0	male	scalp	vidir_modern
4	HAM_0001466	ISIC_0031633	bkl	histo	75.0	male	ear	vidir_modern

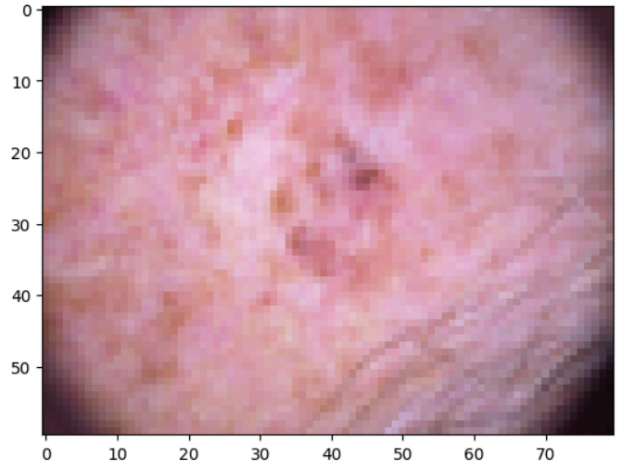
2) **HAM10000_images folder:** This folder contains 10015 gfg image files representing different forms of skin cancer suffered from patients, with the image size being There are a total of 7 classes of skin cancer diseases the patients suffer from. They are as follows:-
    - Bkl: Benign Keratosis lesions
    - Nv: Melanocytic nevi
    - Mel: Melanoma
    - Bcc: Basal cell carcinoma
    - Akeic : Actinic kerotoses
    - Vasc: Vascular lesions
    - Df: Dermatofibra

Originally, HAM10000 was built with the motivation of training neural networks after it was noted that the training of these models were hampered by the small size and lack of diversity of available dermatoscopic images. This dataset is currently available at the Harvard Dataverse page: Harvard dataverse

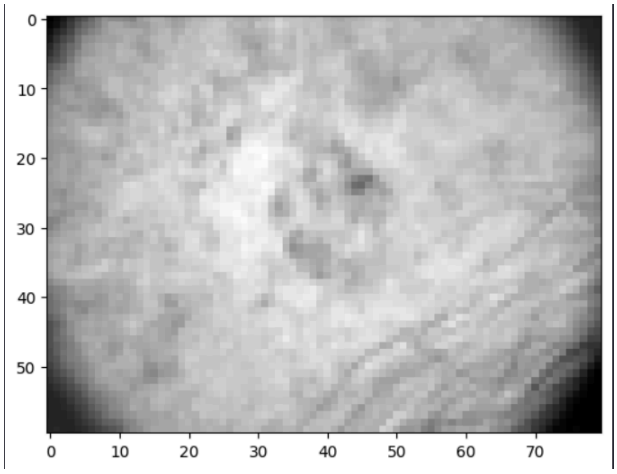
4. Dataset analysis and visualization:

For the given dataset, a total of 5 features were used including the images. The value that we try to predict is the cell type or the ‘dx’ column within our dataset, which tells us what type of skin cancer is seen within the patient
The 4 features selected from the tabular dataset were age, sex, df_type(how the data has been captured) and localization(the area of the body where the cancer has been detected). In case of the image data captured, it has been resized from 600*450 dimensions to 60*80 dimensions and then grayscale for our machine learning models.

Original image

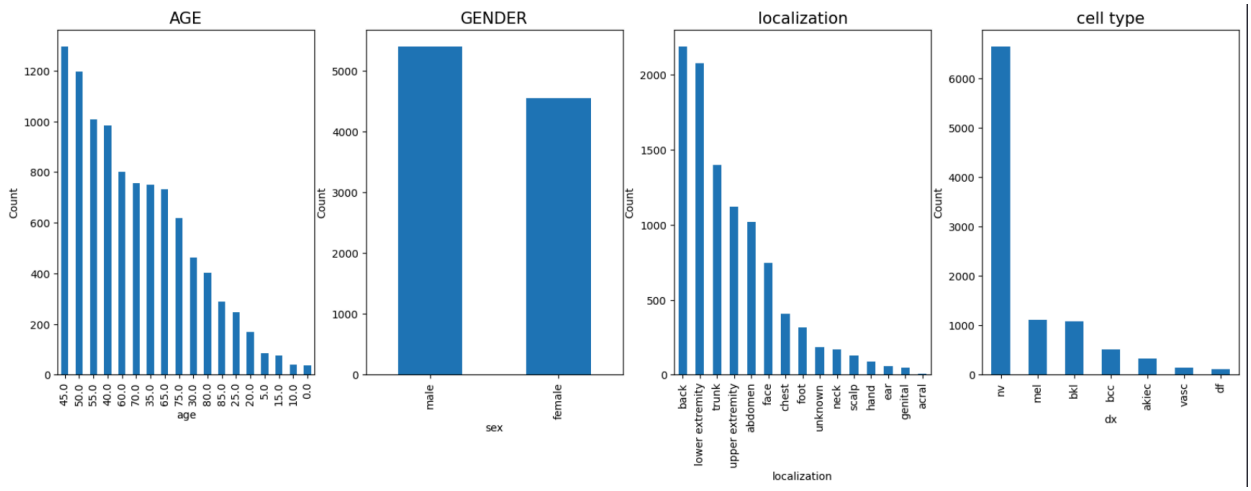


Grayscaled image

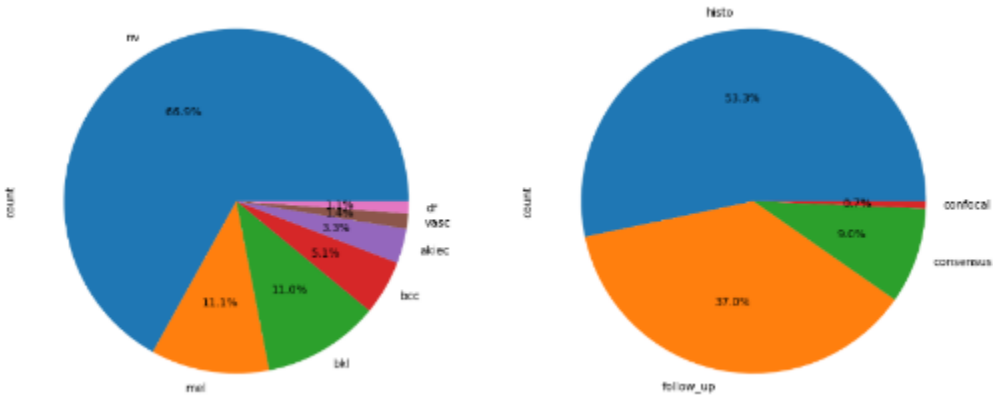


The image data captured is flattened for further dimensionality reduction if necessary in the upcoming steps.

With respect to the tabular dataset, plots were presented and analyzed for the unique values within each feature/column of the dataset. The interesting thing to note in the plots is that the cell type is highly biased towards the class ‘nv’. And there are hardly any cases noticed for the class ‘vasc’ and ‘df’. The difference between the instances of each class gives us a case of class imbalance. Methods to deal with class imbalance is provided in the upcoming sections.

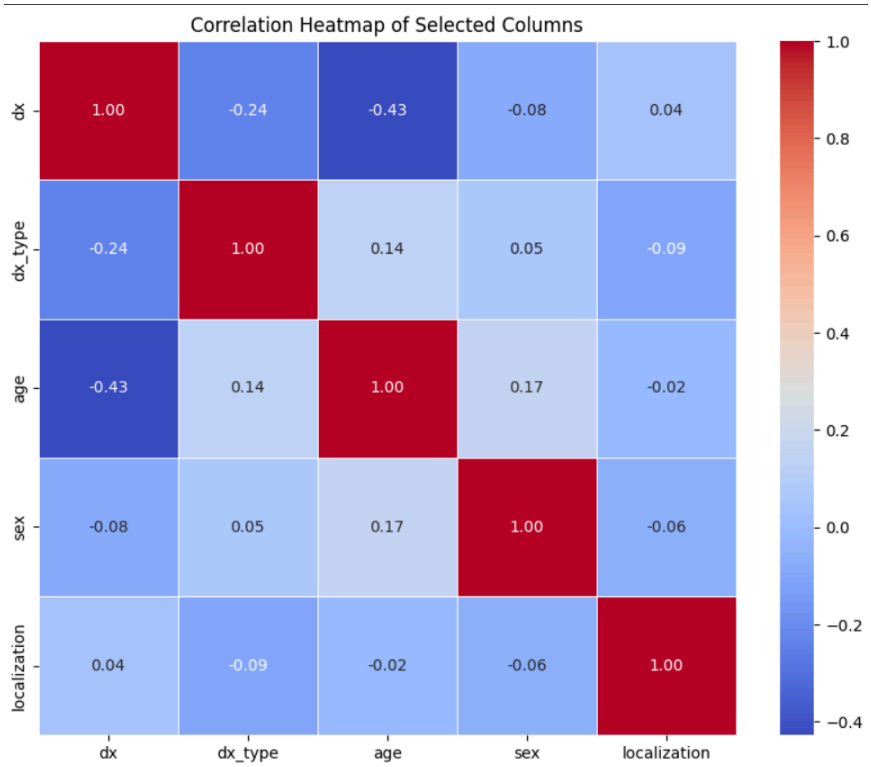


The pie chart gives a dataset description of the cell_type and how each skin cancer case was



detected on the medical end

The correlation matrix of the tabular data and the prediction class has been presented below:-



A higher negative correlation implies that for higher values of a feature space, a lower value of the prediction class would be favored.

5. Data Manipulation:

For the images which were grayscale earlier, each pixel of the images were taken, and a column was assigned to each pixel. The dataset was then scaled down using Standard Scaler. Standard scaler improves the analysis of our models by bringing all the features to one scale and preserving the interpretability and distribution of the original dataset.

To reduce the dimensions used for training of our design and reduce the computational cost of our training model, three methods were used- (a) PCA(supervised), (b) Auto-encoders. For both the parts, the number of dimensions of the image samples were reduced to 100, and sent for data analysis.

To deal with the class imbalance issues, sampling methods were used, with the motive of somewhat preserving the distribution of the original dataset. For the oversampling of the minority class, SMOTE algorithm was used, with the sampling strategy,

```
sampling_strategy = {0: 800, 1:1000, 2: 1500, 3: 400, 4: 1500, 6: 700}
```

And for the majority class, we have used RandomUnderSampling, with the following sampling strategy.

```
desired_count_class_5 = 2000

sampling_strategy = {5: desired_count_class_5}
```

5. Training Models with PCA based dataset:

For the initial analysis, we trained the models using the features we received after running PCA. The performance metrics used for our models are:-

1) Accuracy
2) Precision
3) Recall
4) F1-score

F1-score is a preferable measure for class imbalance as compared to accuracy. A good F1 score indicates that the model performs well in terms of both precision and recall, which is particularly important in scenarios where class distribution is uneven. On the other hand, low accuracy suggests that the model's overall performance in terms of correct predictions is suboptimal, but it doesn't provide insights into the balance between true positives, true negatives, false positives, and false negatives, which F1 score does.

The model evaluations are shown below:-

Model	Accuracy	Precision	Recall	F1-score
RandomForestClassifier	0.7093373493975904	0.4832943777949749	0.32785747294994805	0.3228025630590471
AdaBoostClassifier	0.4703815261044177	0.1424520352605652	0.23331907613344738	0.12927508298533033
SVC	0.4623493975903614	0.2925663815221195	0.39299353961738553	0.2954170360090226
KNeighborsClassifier	0.6797188755020008	0.3133109237668304	0.23505546207428157	0.24668566356004845
LogisticRegression	0.7218875502008032	0.3677086358896945	0.28020054605597805	0.2952554670062323
MLPClassifier	0.6997991967871486	0.3826510351470617	0.35995891217977906	0.3683684130901567
GaussianNB	0.5251004016064257	0.2667708989323765	0.35597862864285568	0.2631498147845606
DecisionTreeClassifier	0.6696787148594378	0.3189497416920138	0.32715641075704205	0.3206112216923156
GradientBoostingClassifier	0.7454819277108434	0.5118907481776286	0.36091187282051186	0.4024894163852624
XGBClassifier	0.7319277108433735	0.4981967435581846	0.42168915277584634	0.4342457339614958

6. Cross Fold Validation:

Based on the analysis of our models, we have also done cross fold validation, with K_folds=5, that is 80% of the training samples were used for training and the rest was used for validation keeping the same model metrics. The results of the model are shown below:-

Model	Accuracy	Precision	Recall	F1-score
XGBClassifier	0.7279116465863453	0.4762648594209037 4	0.4180198195427834 4	0.4252296636414382 7
GradientBoostingClassifier	0.6651606425702812	0.4006359156062087 6	0.4888659219571287	0.4275258678473776
MLPClassifier	0.5868473895582329	0.3071617289912486	0.3380889308555159	0.3150389397082547 5
RandomForestClassifier	0.6470883534136547	0.4787955391018297	0.4791688394624885 6	0.4313901676826 41

7. Model analysis with autoencoders:

For the next segment, instead of the dataset which used PCA, we use a dataset whose dimensions were reduced through unsupervised learning methods i.e. using autoencoders. The autoencoders used a total of 8 hidden layers and the aim was to reduce the number of dimensions to 100.

Based on that the following results were obtained, for two models which performed well initially:-

Model	Accuracy	Precision	Recall	F1-score
Grad. Boost	0.6265060240963856	0.2789445747622749	0.2978680281761311 7	0.2845122526770196
XG Boost	0.6380522088353414	0.2934261845772578	0.3162017016869382	0.3016598679961654 5

As noticed in this case, the models trained on data samples derived out of autoencoders perform poorly as compared to the ones derived from PCA.

8. Analysis using CNNs:

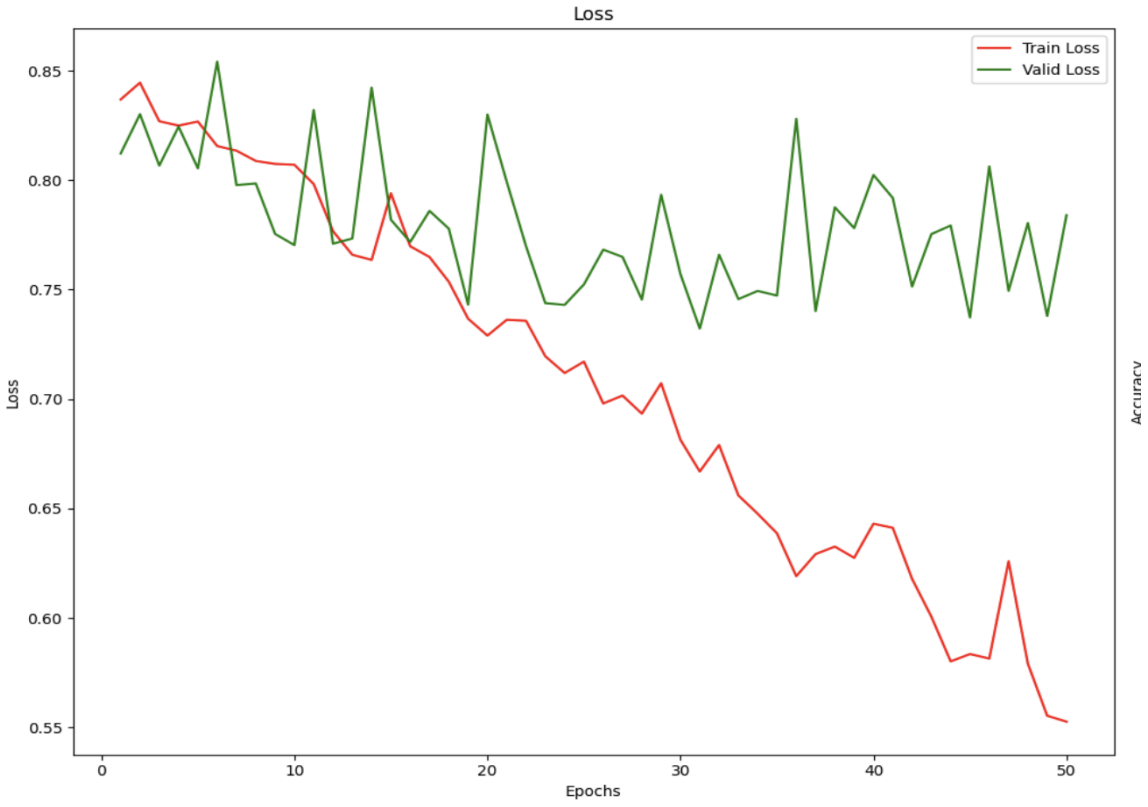
For the final analysis, we use CNNs or Convolutional Neural Networks where only the image data has been used to train the model.
The following architecture was used for the model:-

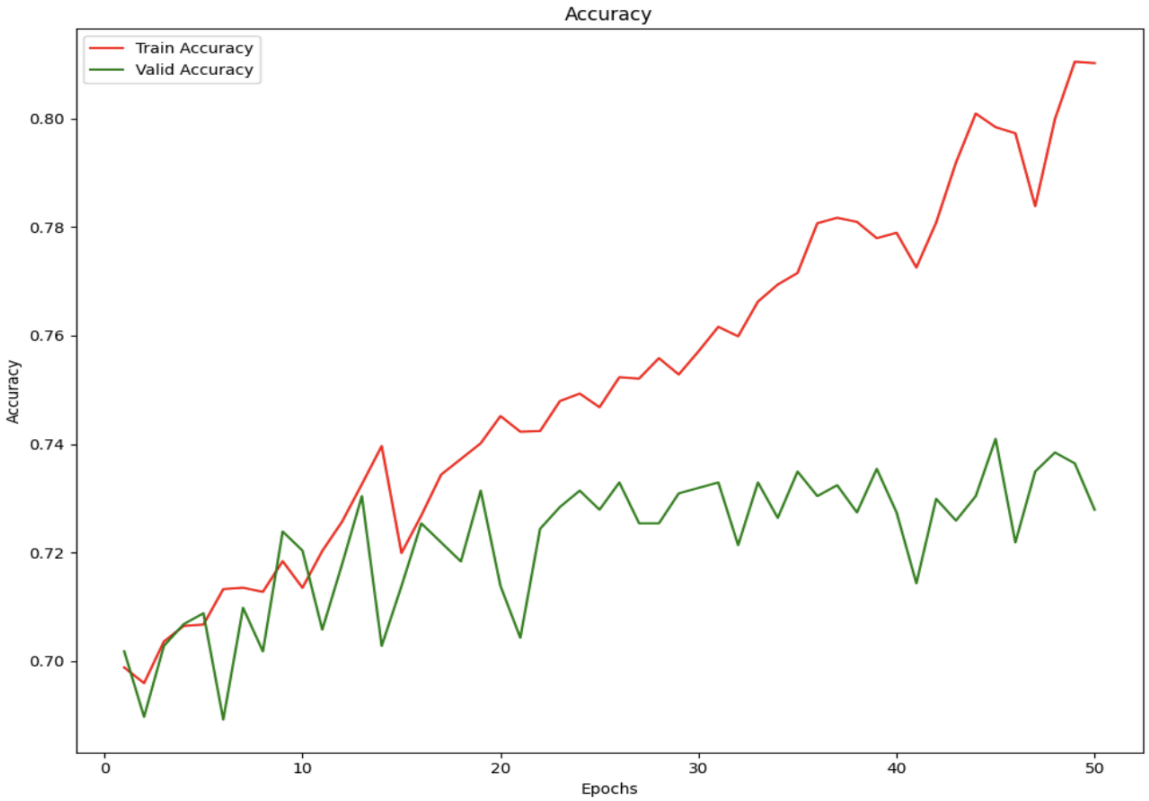
```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(60, 80, 3)),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dense(7, activation='softmax')
])

model.add(Dense(units=7, activation='softmax', name='classifier'))
```

Based on that the following results were obtained:

Loss	Accuracy	Precision	Recall	F1-score
0.7839683294296265	0.7279116511344911	0.7762863534675615	0.6967871485943775	0.7343915343915344

Graphs for accuracy and loss:




Accuracy
Train Accuracy
Valid Accuracy
0.80
0.78
0.76
0.74
0.72
0.70
Accuracy
0
10
20
30
40
50
Epochs