Assignment 2: Machine Learning Project: Classify Images of Cancer

Introduction and Goal: The goal is to create a machine-learning system that can categorize images of malignant cells from histopathology. A modified version of the "CRCHistoPhenotypes" dataset is used for this challenge. The 27x27 RGB photographs of cells from 99 different patients make up the dataset, which will be used for two tasks: Cancerous/ not Cancerous and Cancer cell type.

Binary Classification – is Cancerous/ is not Cancerous.

Exploratory Data Analysis: Inferences from the EDA are as follows. We merged both the datasets (extra data, main data) to produce a data frame. It was found that there was an imbalance in the classes – Cancerous/ Not Cancerous. (0,1). It was also found that, the data related to a few patients was not distributed evenly.

Baseline Model (DATASET used: data_labels_ExtraData) The base model defined has a Flatten layer as the first layer which flattens the input tensor to a 1D tensor. This is followed by a Dense layer with 256 units, and another Dense layer with 2 units, which is the output layer with a sigmoid activation function, SDG optimizer, loss as binary cross entropy and used accuracy metrics for binary classification. Splitting the data into 70% train, 20% validation, and 10% test sets. Using the flowfromdataframe, the generator rescales the pixel values of the images by a factor of 1./255 to normalize them to the range [0, 1]. The val_datagen, train_datagen, test_datagen generator applies only the rescaling transformation to the data.

From the results, we can observe that the training accuracy is higher than the validation accuracy, which suggests that the model might be **overfitting** the training data due to class imbalance and this indicates that it requires regularization techniques or more data to improve its performance. Therefore, the model has a **high variance.** However, the model has a good bias, since the training and validation metrics are relatively close.

Custom Model Description (DATASET used: data_labels_ExtraData and data_labels_MainData) As we used only one dataset earlier, we have merged both datasets now to provide more data to the model and performed data augmentation to balance the classes in the dataset.

Data Augmentation: Read each image, resize it to the desired input size, and perform data augmentation using the specified data augmentation parameters like rotation range, zoom range, horizontal and vertical flips, and fill mode. The code splits the original dataset into two DataFrames based on the class label (cancerous or non-cancerous). It then determines the class with fewer samples and its corresponding data frame. The augmented DataFrame is concatenated with the original majority class DataFrame to create a balanced DataFrame to avoid class imbalance, thereby improve the generalization of the model.

Model Description: The input image shape is (27, 27, 3), and the first layer is a 2D convolutional layer with 16 filters of size 3x3 and a ReLU activation function. Then, a max pooling layer with a pool size of 2x2 is added, which reduces the spatial dimensions of the feature maps. The same pattern is repeated for two more convolutional layers with 32 and 64 filters, respectively. Afterward, the output from the convolutional layers is flattened and fed to a dense layer with 64 units and a ReLU activation function, followed by another dropout layer. Finally, the output layer with a sigmoid activation function. The model is trained using binary cross-entropy loss and the Adam optimizer, and accuracy is used as the evaluation metric.

From the results, The model performed well in detecting cancerous cells with high accuracy, precision, recall, and specificity. However, the slight difference in performance between the test set and the validation and train sets suggests that the model may have some variance issues and could benefit from further optimization. The model achieved an accuracy of 88.31%, precision ranging from 82.46% to 90.39%, recall of 84.44%, and F1 score of 83.44% on the test set. The validation set had an accuracy of 87.93%, precision of 91.14%, recall of 84.91%, specificity of 91.17%, and F1 score of 87.92%. The training set metrics were comparable to the validation set metrics, and the ROC AUC scores were high for all three sets. Overall, the second model showed good performance in detecting cancerous cells.

Keras Tuner Model (DATASET used: data_labels_ExtraData and data_labels_MainData) The Keras tuner aims to maximize the validation accuracy by conducting up to 10 trials and employing early stopping based on the validation accuracy metric. The model architecture is designed to be flexible, allowing customization of the number of layers and

their respective hyperparameters. The activation functions, such as ReLU, introduce non-linearity to the model, enabling it to learn complex patterns and make accurate predictions. The convolutional layers perform 2D convolutions on the input images, extracting relevant features through tunable filters. Max pooling is applied after each convolutional layer to reduce the spatial dimensions of the feature maps. The dense layers capture intricate relationships between features, and dropout regularization is implemented to prevent overfitting. The output layer utilizes a sigmoid activation function for binary classification. The model is compiled with a stochastic gradient descent optimizer, with a tunable learning rate, and the binary crossentropy loss function is used to measure the difference between predicted and true class labels. The evaluation metric employed is accuracy, which measures performance.

From the results, the Keras Tuner model has relatively **low bias and variance**. This is indicated by the fact that the accuracy, precision, recall, specificity, and F1 score are all consistent across the test, validation, and train datasets, with a relatively small difference between the performance on the train and validation sets. The model has achieved an accuracy of 0.8462 on the Test dataset, 0.8564 on the Validation dataset, and 0.8583 on the Train dataset. Precision, recall, specificity, and F1 Score are also reported for each dataset. The Validation dataset has the highest precision of 0.8576, while the Train dataset has the highest recall of 0.8900. The F1 Score is the highest on the Train dataset with a score of 0.8666.

Multi-Class Classification – cell Type (fibroblast, inflammatory, epithelial, or others).

Exploratory Data Analysis: Inferences from the EDA are as follows. We used the dataset - main data to produce a data frame. It was found that there was an imbalance in the classes –cell types. (0,1,2,3). It was also found that, the data related to a few patients was not distributed evenly.

Base Model: (DATASET used: data_labels_MainData) defined has a Flatten layer as the first layer which flattens the input tensor to a 1D tensor. This is followed by a Dense layer with 256 units, and another Dense layer with 2 units, which is the output layer with a SoftMax activation function for binary classification.

From the results, the model has a **high bias**, which means that the model is unable to capture the underlying patterns in the data correctly. Additionally, the **variance of the model is relatively low**, which suggests that the model is not overfitting on the training data, but it is not generalizing well to the test and validation data. While the low variance is desirable, it will not benefit the accuracy of the model if the **bias remains high.** The overall accuracy of the model was 32%. The weighted precision, recall, and F1-score were 0.29, 0.32, and 0.30, respectively. The results indicate that the model has poor performance in distinguishing between the four classes.

Model: Keras Tuner: (DATASET used: data_labels_MainData) As we used the imbalanced dataset earlier, we have performed data augmentation to balance the classes in the dataset.

The function builds a CNN with tuneable hyperparameters such as the number of convolutional layers, the number of filters, the kernel size, the activation function, the number of dense layers, the number of units in each dense layer, the learning rate, and more. The model incorporates convolutional layers with ReLU activation and max pooling to capture relevant features and reduce spatial dimensions. Dense layers with ReLU activation are added for learning complex relationships between features. Dropout regularization is applied to prevent overfitting. The final output layer uses SoftMax activation for multi-class classification. Furthermore, the weights of the convolutional layers are initialized using a random normal distribution, while biases are set to zero. The RandomSearch object is then instantiated with the build_model function, the objective function to optimize (val_loss in this case), the maximum number of trials to run (max_trials), and the directory and project name to save the results. The EarlyStopping object is defined to monitor the validation loss during the hyperparameter search and stop the search if the validation loss stops improving by more than min_delta after a patience number of epochs.

Data Augmentation: address the class imbalance in a dataset by generating additional images for the minority classes. The function first creates an ImageDataGenerator object from the Keras library, which performs the types of data augmentation. Then, for each image in the DataFrame, the function loads the image, applies the specified data augmentation using the ImageDataGenerator, and saves the augmented image(s) to the output directory. The code then moves the generated images into the main image folder. The code repeats this process multiple times, each time augmenting a different DataFrame (class0 – 4 times, class1 – 3 times, class2 – 2 times, class3 – 5 times) and then concatenating the resulting augmented DataFrame with the original DataFrame for that class.

From the results, **the bias of the model seems good** as indicated by the higher accuracy on the test, training, and validation datasets. The accuracy suggests that the model is capturing a reasonable amount of information from the data and has a relatively lower bias. The model's evaluation results show an accuracy of 75.05% on the test set, 72.12% on the training set, and 70.81% on the validation set. The evaluation results indicate that the model's accuracy on the training data is slightly higher compared to the test data and validation data. This suggests a **moderately low variance**, meaning the model is not overfitting the training data excessively.

VGG 16: : (DATASET used: data_labels_MainData) The top layers of the VGG16 model are frozen and several custom layers are added on top with L2 regularization and dropout. The model is compiled with categorical cross-entropy loss and the Adam optimizer, and accuracy is used as the evaluation metric. Early stopping and learning rate reduction on plateau are used as call-backs during training. The custom layers introduced, implement non-linearity and help the model learn more complex representations. **The L2 regularization and dropout layers added to the model help prevent overfitting during training.**

Data Augmentation: The **train_datagen** generator applies various image transformations to the training data, such as rotation, width and height shifting, zooming, and horizontal flipping. These transformations are intended to augment the training data and increase its diversity, which can help improve the model's ability to generalize to new data. Additionally, the generator rescales the pixel values of the images by a factor of 1./255 to normalize them to the range [0, 1]. The **val_datagen** and **test_datagen** generator applies only the rescaling transformation to the data. This is because we typically do not want to augment the validation and test data since we want it to be a true representation of real-world data.

From the results, the model has performed reasonably well on the training, validation, and test sets with an a good accuracy. The model was able to achieve an accuracy of 77% on the training set and 75% on the testing set. In this case, since the accuracy on all sets is relatively close, it suggests that the model has generalized well and there is **no significant bias or variance issue.**

Using Extra Dataset: data_labels_extraData to increase the performance of the model (DATASET used: data_labels_ExtraData and data_labels_MainData) We used extra data to predict the cell type and used that data to train our hyperparameter-tuned model and tested the model on unseen data. In summary, the code uses a trained model to make predictions on images of extra dataset. It compares the predicted probabilities with the median values of correctly classified instances and assigns the new data to the class with the closest median value. The results are stored in a data frame. This data frame was used earlier used train our model, the F1 scores of the predicted class did not improve compared to the earlier model's result.

Ultimate Judgement:

Task 1: Cancerous/ Not Cancerous:

The baseline model, and it shows some issues with overfitting, as the training accuracy is higher than the validation accuracy. The custom model and there is a slight difference in performance between the test set and the validation and train sets, it is within acceptable limits. The Keras Tuner model has relatively **low bias and variance**. The model shows consistent performance across the test, validation, and train datasets, with a relatively small difference in performance between the training and validation sets. The model has achieved high accuracy, precision, recall, specificity, and F1 scores across all three sets. Therefore, this model is a good choice.

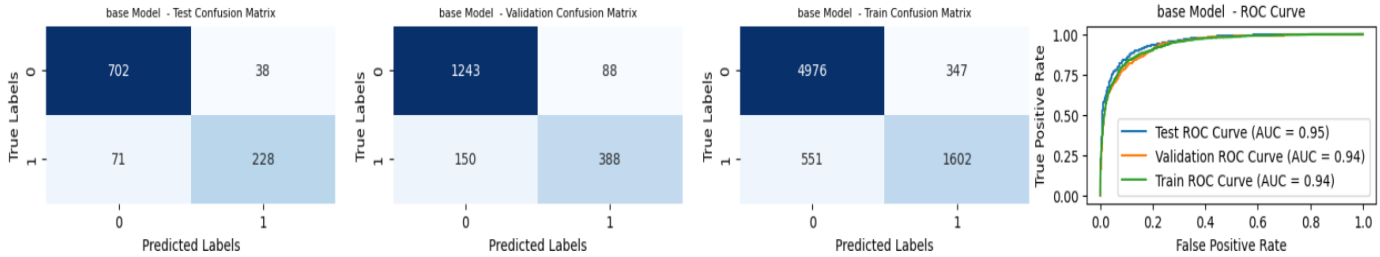
Task 2: cellType - Cancer:

The baseline model has a high bias and low variance, indicating that it is unable to capture the underlying patterns in the data correctly. On the other hand, the Keras Tuner model has achieved a relatively higher accuracy on the test, validation, and training datasets, indicating that the model is capturing a reasonable amount of information from the data and has a relatively **lower bias and variance**. The VGG16 model has performed reasonably well on all sets, achieving a good accuracy, and suggesting that the model has generalized well**, without significant bias or variance issues**. Regarding the use of extra data to increase the performance of the model, the F1 scores of the predicted class did not improve compared to the earlier model's results. In conclusion, the Keras Tuner model and VGG16 model appear to perform well in distinguishing between the cell types.

Appendix – Task 1

ROC curve and metrics – Baseline Model

```
      base Model -Test,Validation and Train Metrics
Test Model: base Model              Validation: base Model              Train: base Model
Test Accuracy: 0.8951               Validation Accuracy: 0.8727         Train Accuracy: 0.8799
Test Precision: 0.8571              Validation Precision: 0.8151        Train Precision: 0.8220
Test Recall: 0.7625                 Validation Recall: 0.7212           Train Recall: 0.7441
Test Specificity: 0.9486            Validation Specificity: 0.9339      Train Specificity: 0.9348
Test F1 Score: 0.8071               Validation F1 Score: 0.7653         Train F1 Score: 0.7811
```



Before and After Data Augmentation – Balancing the classes

```
# cheacming how class is distributed
data['isCancerous'].value_counts()
```

```
0    13211
1     7069
Name: isCancerous, dtype: int64
```

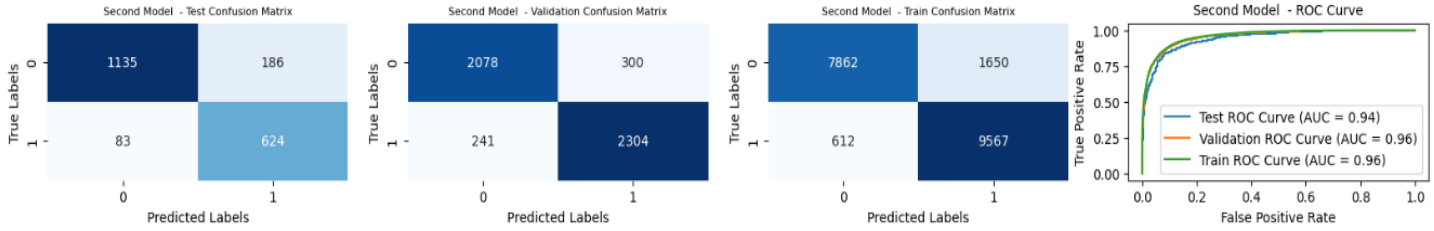
```
# after data augmentation we could incre

train_df['isCancerous'].value_counts()
```

```
1    12724
0    11890
Name: isCancerous, dtype: int64
```

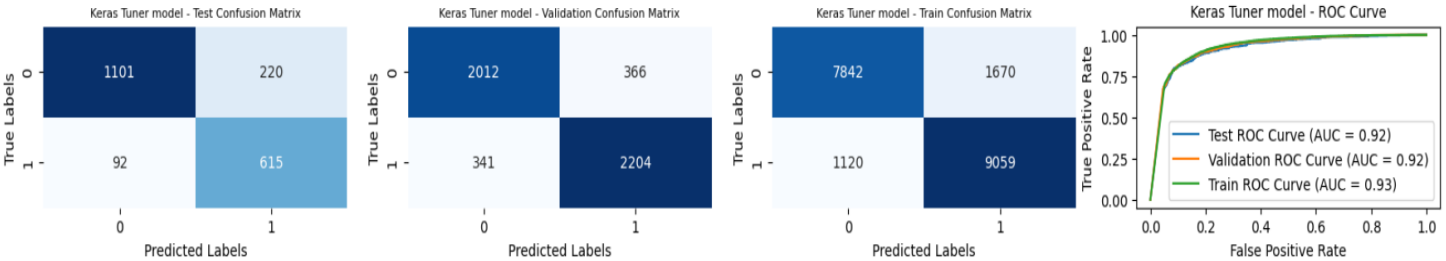
ROC curve and metrics – Custom Model

```
      Second Model -Test,Validation and Train Metrics
Test Model: Second Model            Validation: Second Model            Train: Second Model
Test Accuracy: 0.8674               Validation Accuracy: 0.8901         Train Accuracy: 0.8851
Test Precision: 0.7704              Validation Precision: 0.8848        Train Precision: 0.8529
Test Recall: 0.8826                 Validation Recall: 0.9053           Train Recall: 0.9399
Test Specificity: 0.8592            Validation Specificity: 0.8738      Train Specificity: 0.8265
Test F1 Score: 0.8227               Validation F1 Score: 0.8949         Train F1 Score: 0.8943
```



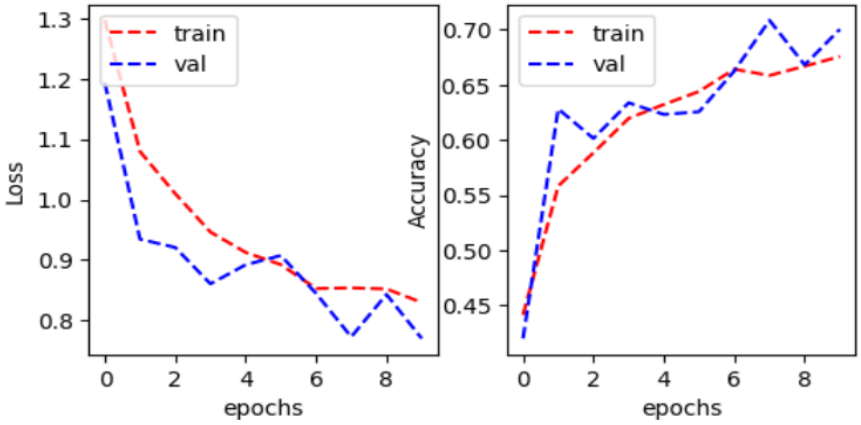
ROC curve and metrics – Keras Tuner (HyperParameter Tuning)

```
   Keras Tuner model-Test,Validation and Train Metrics
Test Model: Keras Tuner model       Validation: Keras Tuner model       Train: Keras Tuner model
Test Accuracy: 0.8462               Validation Accuracy: 0.8564         Train Accuracy: 0.8583
Test Precision: 0.7365              Validation Precision: 0.8576        Train Precision: 0.8443
Test Recall: 0.8699                 Validation Recall: 0.8660           Train Recall: 0.8900
Test Specificity: 0.8335            Validation Specificity: 0.8461      Train Specificity: 0.8244
Test F1 Score: 0.7977               Validation F1 Score: 0.8618         Train F1 Score: 0.8666
```



Appendix – Task 2

Learning Curve – BaseLine Model



Before and after Data Augmentation (balancing the classes)

```
data_main['cellType'].value_counts()
```

```
2    4079
1    2543
0    1888
3    1386
Name: cellType, dtype: int64
```

```
merged_df['cellType'].value_counts()
```

```
2    7342
1    6867
0    6796
3    6235
Name: cellType, dtype: int64
```

Learning Curve – VGG16 Model