Fruits 360

Ramanpreet - B20CS052
Swara Patel - B20AI044

Abstract –

This *report showcases our experience with building a classifier that can distinguish between 131 categories of Fruits and Vegetables combined. We have a dataset that consists of feature vectors of images of fruits rotated at different angles each of 100*100 dimensions. We use various classification algorithms and compare their results in this report.*

I. INTRODUCTION

There is a wide variety of fruits and vegetables available, sometimes it becomes difficult for humans to distinguish between such a wide range of varieties. Fruit and vegetable classification has a wide range of uses, including automated harvesting by robots, stockpiling for supermarkets, effective detection of specific flaws, and assessing fruit ripening. Thus distinguishing between fruits and vegetables becomes an important problem to solve. This report aims to showcase the results of different machine learning algorithms i.e. SVM, Random Forest, KNN & LightGBM in classifying fruits and vegetables. Apart from conventional models, this report also includes results of Convolution Neural Networks, Feature Extraction using AutoEncoders, and Transfer Learning on pre-trained VGG16 and ResNet models to achieve high accuracy.

Datasets

Fruits 360: The folder Fruits 360 available on Kaggle is used as a dataset.
Labels: 131 (fruits + vegetables)
Number of samples in Train dataset: 67692 (one fruit/vegetable per image)
Number of samples in Test dataset: 22688 (one fruit/vegetable per image)
Image Size: 100 x 100 (pix)

II. METHODOLOGY

Overview

There are various classification algorithms present out of which we shall implement the following

- *Random Forest Classification*
- *KNN*
- *LightGBM*
- *SVM*
- *CNN*
- Use of AutoEncoders for feature extraction
- Transfer Learning on pre-trained VGG16 and ResNet50 models

Exploring the dataset and pre-processing

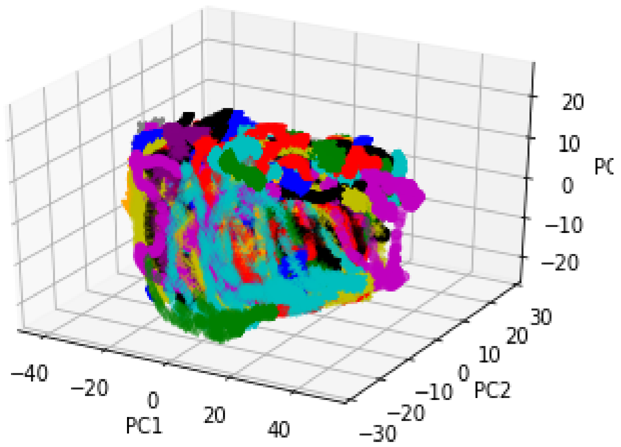
There were no NULL values/empty files found in the dataset folder. Using the OpenCV-python library, the images are read to a NumPy array (the dimensions of the images stored in the dataset are 20 x 20) and a dataset is created. Next, the image samples in datasets are flattened to a 1D array so as to form a row in the data frame. No further preprocessing was required. For Transfer learning, the images were used with their original dimensions (i.e. 100 x 100) and loaded batch by batch directly from the folder using TensorFlow and Keras.

Visualizing the datasets

Training Set

Testing Set

Principal Component Analysis (n_components=3)



Implementation of classification algorithms

- *Support Vector Machine:* In SVM, data points are plotted into n-dimensional graphs which are then classified by drawing hyperplanes.
  - A simple SVC with a linear kernel was implemented
  - (RBF kernel was taking more than 2 hours to fit, proving it to be too complex, so we didn't include it in our reports due to complexity reasons)

- *KNN (k - nearest neighbors):* KNN are supervised algorithms that classify on the basis of distance from similar points. Here k is the number of nearest neighbors to be considered in the majority voting process.
  - Three types of KNN Classifiers were implemented:
    - KNN with n_neighbors = 5
    - KNN with n_neighbors = 15

        - KNN with n_neighbors = 25

- *Random Forest*: Random Forest Classifiers use boosting ensemble methods to train upon various decision trees and produce aggregated results. It is one of the most used machine learning algorithms.
    - Two types of Random Forest Classifiers were implemented:
        - RF without hyperparameter tuning
        - RF with hyperparameter tuning

- *LightGBM:* LightGBM is a gradient boosting framework based on Decision Trees to increase the efficiency of the model and reduce the memory usage.
    - Two types of LightGBM Classifiers were implemented:
        - LGB without hyperparameter tuning
        - LGB with hyperparameter tuning

- Convolutional Neural Networks: CNN is a type of deep neural network that extracts meaningful information/patterns from the input features which is further used to build subsequent layers of neural network computations by making use of kernels.
    - A simple 4 layered CNN was implemented
        - ReLu as activation function - applied after every convolution to transform the output values to the range (0,1)
        - Max Pooling - to downsample the input representation, helps deal with overfitting

- *Feature Extraction using AutoEncoder*: AutoEncoder is a feed-forward kind of neural network where input is the same as output. It consists of Encoder, Code & Decoder. They are used for feature extraction i.e. dimensionality reduction of images. Extracted features are used as input to the desired classifier models.
    - Encoder
        - A MaxPool layer sandwiched between 2 blocks of each consisting of a Convolutional Layer, an activation layer with ReLu function, and a Batch Norm Layer
    - Middle Layer
        - A MaxPool layer followed by a block consisting of a Convolutional Layer, an activation layer with ReLu function, and a Batch Norm Layer
    - Decoder
        - 2 blocks each consisting of a Convolutional Layer, an activation layer with ReLu function, and a Batch Norm Layer each followed by Upsampling
    - Output
        - A convolutional Layer followed by an Activation layer with ReLu function

  SVC-linear and KNN-5 are trained after the feature extraction from autoencoder.

- *Transfer Learning* - Transfer learning involves the use of pre-trained models where the topmost layer is removed and the remaining weights are frozen, the model only needs to learn the weights of the new output layer which predicts the class

    - VGG16 -
        - Epochs = 10
        - Optimizer = Adam(lr = 0.001)
        - Loss = Categorical cross-entropy
        - Pooling = Average
        - New 2 layers - with activation functions ReLu and softmax respectively
    - ResNet-
        - Epochs = 10
        - Optimizer = Adam(lr = 0.001)
        - Loss = Categorical cross-entropy
        - Pooling = Average
        - New 2 layers - with activation functions ReLu and softmax respectively

    - After performing transfer learning i.e. training on a model with frozen weights, to perform hyperparameter tuning the weights are now unfrozen and the model is trained with a very low learning rate - this was performed for both of the above model

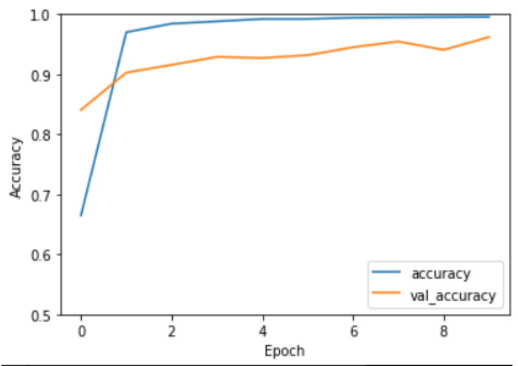
III. EVALUATION OF MODELS

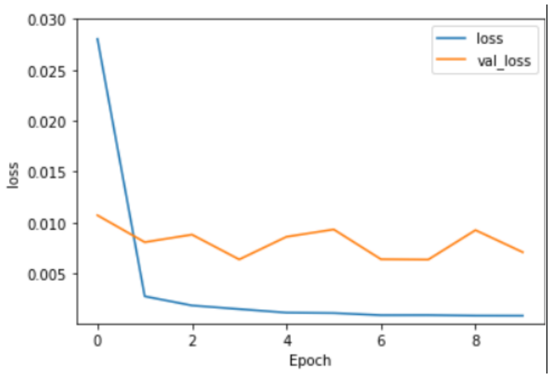
The models implemented were evaluated using techniques like - *accuracy_score, f1_score, roc_auc_score, classification report*:

	Train		*Test*		
	accuracy_score	*f1_score*	*accuracy_score*	*f1_score*	*ROC Value*
SVC - linear	*0.99945*	*0.99944*	*0.83441*	*0.82676*	*0.91366*
KNN - 5	*0.99899*	*0.99894*	*0.87892*	*0.87473*	*0.93688*
KNN - 15	*0.99419*	*0.99382*	*0.83907*	*0.83312*	*0.91648*
KNN - 25	*0.98413*	*0.98331*	*0.80642*	*0.79988*	*0.90038*
RF	*1.0000*	*1.0000*	*0.90148*	*0.89460*	*0.94838*
RF - tuned	*1.0000*	*1.0000*	*0.94895*	*0.94525*	*0.97248*
LGB	*0.30485*	*0.29987*	*0.24444*	*0.23555*	*0.52410*
LGB - tuned	*0.44257*	*0.44205*	*0.34784*	*0.34404*	*0.66789*
CNN	*0.9957*	*-*	*0.9620*	*-*	*-*
SVC - linear (autoencoder as input)	*1.0000*	*1.0000*	*0.9516*	*0.94743*	*0.97371*
KNN - 5 (autoencoder as input)	*0.99905*	*0.99900*	*0.877864*	*0.87364*	*0.93631*
VGG16	*0.9958*	*-*	*0.9448*	*-*	*-*
VGG16 - tuned	*1.000*	*-*	*0.9813*	*-*	*-*
ResNet50	*0.9979*	*-*	*0.9463*	*-*	*-*
ResNet50 - tuned	*0.9993*	*-*	*0.9842*	*-*	*-*

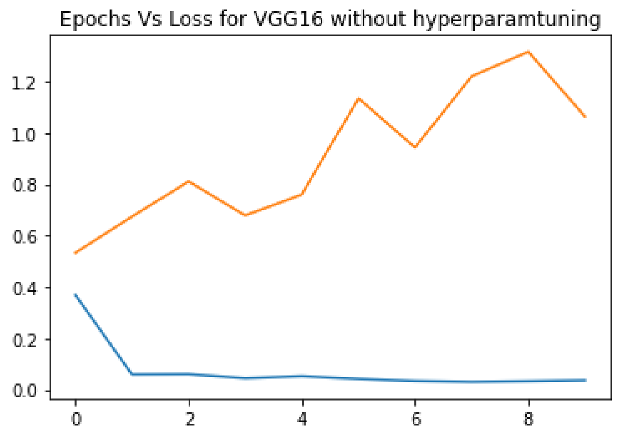
ROC - Plots :
Since this is a multiclass classification problem, while ROC_AUC is defined for binary classification problems, so if graphs were to be plotted, 131 graphs would be required for each model belonging to each glass, which requires a lot of memory, hence didn't make any sense to plot it. Instead, we included ROC_AUC_Score by OneHotEncoding the y_test and y_pred values and getting ROC_AUC_Score on it.

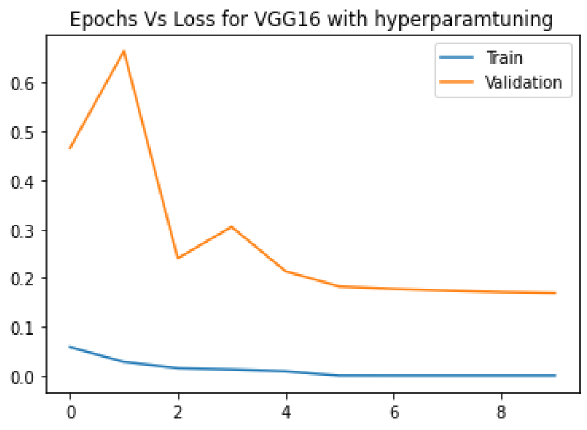
CNN Graphs:

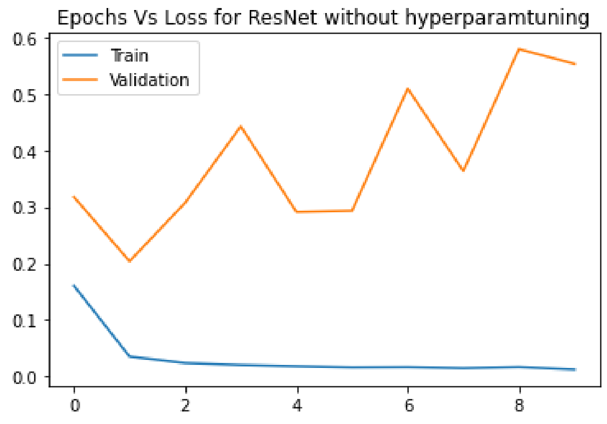


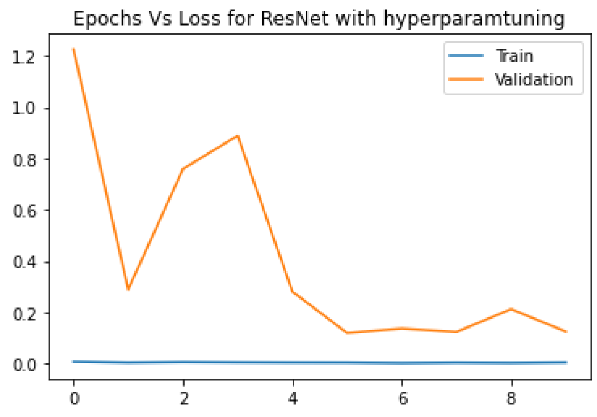


Transfer Learning Graphs:









IV. RESULTS AND ANALYSIS

From the table, it can be concluded that LGBM performed the worst, there was a kind of underfitting. SVM and KNN-15 & KNN-25 gave almost similar results.KNN-5 performed better than the other two versions of KNN. Random Forest performed very well with ROC_AUC_Score equal to 0.9724. The SVC model with features extracted by an autoencoder as input outperforms all models with ROC _AUC_Score=*0.97371*. Similarly the performance of KNN-5 model is highly improved with autoencoder This is due to the fact that autoencoders extract important features from the data. Next, other than these conventional models, CNN performed really well with 96% accuracy on test data. And Transfer learning performed the best among all the classification techniques as expected. ResNet with an accuracy of 0.984 on the test performed slightly better than VGG16 with an accuracy of 0.981 on the test set. The reason transfer learning gives such good results is it helps in fast feature extraction since it is already pre trained on a vast dataset, the learning gained while training over that dataset is used here where we have fewer data available.

CONTRIBUTIONS

The learning and planning was done as a team. The individual contributions are as given

- Ramanpreet (B20CS052): SVM, KNN, LGBM, RF, CNN, AutoEncoder, Report
- Swara Patel (B20AI044): SVM, KNN, LGBM, RF, Transfer Learning, Report

REFERENCES

- https://www.kaggle.com/datasets/moltean/fruits
- https://keras.io/guides/transfer_learning/
- https://keras.io/api/applications/
- https://machinelearningmastery.com/autoencoder-for-classification/