Project Report: Leaf Species Classification

Introduction:

The objective of this project is to create a classification model that can correctly identify various leaf species according to their traits. The 1,584 images of leaf specimens, 16 samples each from 99 different species, make up the dataset used for this project. Three sets of features are included with each image: a shape contiguous description, an interior texture histogram, and a fine-scale margin histogram. Each image in the dataset has a unique ID, and each characteristic has 64 attribute vectors.

Data Preparation:

The data preparation phase involves:

1. importing the required libraries.
2. loading the dataset.
3. Showing data insights:

   The dataset consists of the following columns:

   id: An anonymous ID unique to each image.

   species: The species of the leaf.

   margin1 to margin64: Attribute vectors for the margin feature.

   shape1 to shape64: Attribute vectors for the shape feature.

   texture1 to texture64: Attribute vectors for the texture feature.
4. Checking data types and found all columns are numerical except Species (target): which we can encoding in step ().
5. Cleaning data by checking for nulls and duplicates and outliers ,and we found that there is no null ,no duplicates ,and some outliers but they have minimal impact.
6. Checking number of classes and found them =99.
7. Data Visualization using PCA and drawing some images.
8. Carrying out correlation analysis which leads to : shape_2, shape_3, ..., shape_64 with high correlation A set of highly correlated attributes will add to the algorithm's complexity rather than provide any new information, or very little of it.so we dropped them.
9. Data Division to data and target.
10. Data Standardization using mean and standard deviation.
11. Encoding the labels using Label Encoder.

Training a neural network

1. Splitting the data
2. Building the functions we used as model function, evaluation function, and graphing function.
3. Comparing between Normalized data and Scaled data by running the model for each one and comparing the performance to decide if standardization step is useful or not: which gives us that scaled data achieved higher performance so standardization is valuable for our work. normal data testing accuracy= 0.9499. scaled data testing accuracy= 0.9531.
4. Hyperparameters tunning which we will discuss in detail in the next section.

Hyperparameters

1.Batch Size

We have worked with MBGD and tried different batch sizes (32, 64, 128) for model training and evaluation.

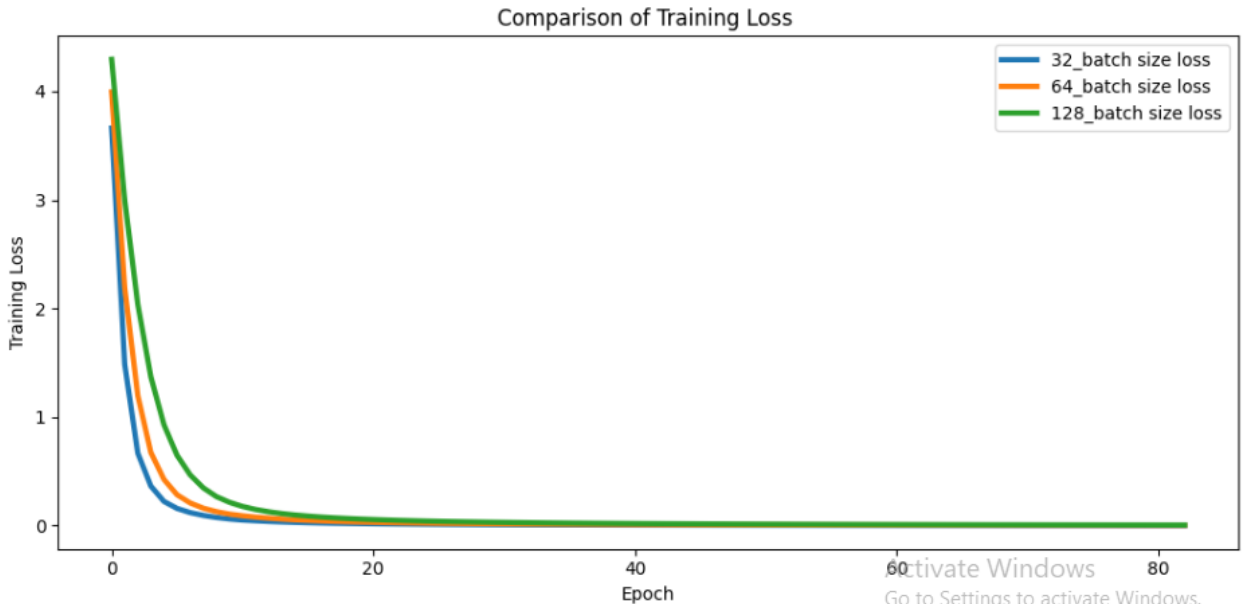
we get that:

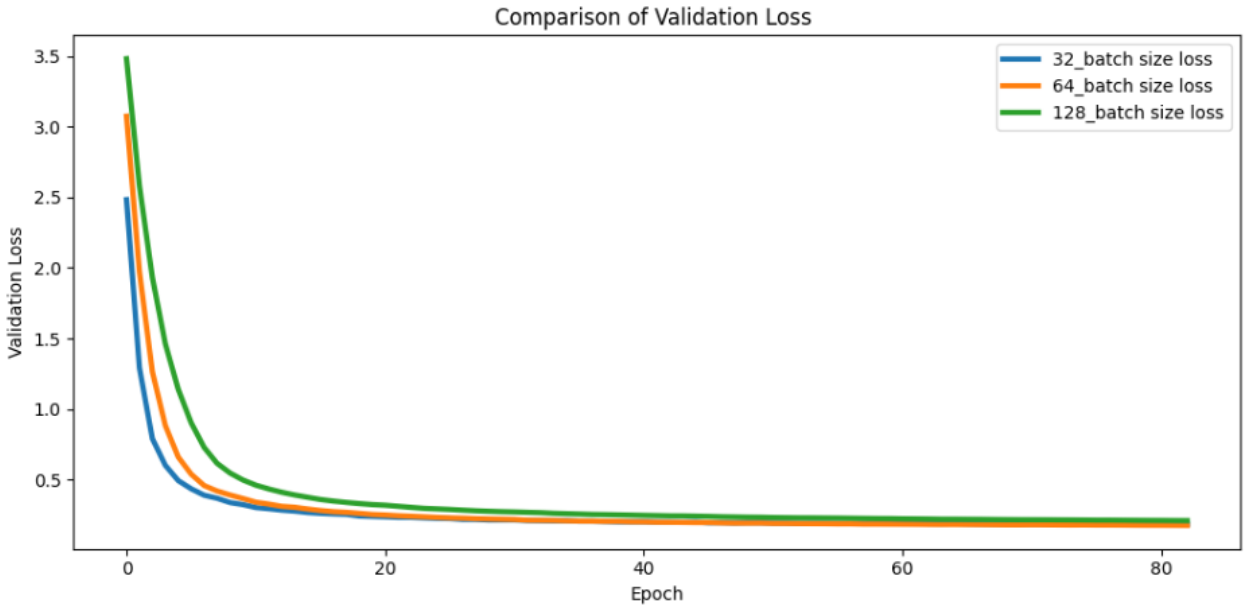
32 batch size achieves highest accuracy and lowest loss very fast as shown in graphs below, but for long term work over the epochs 128 achieved higher accuracy.

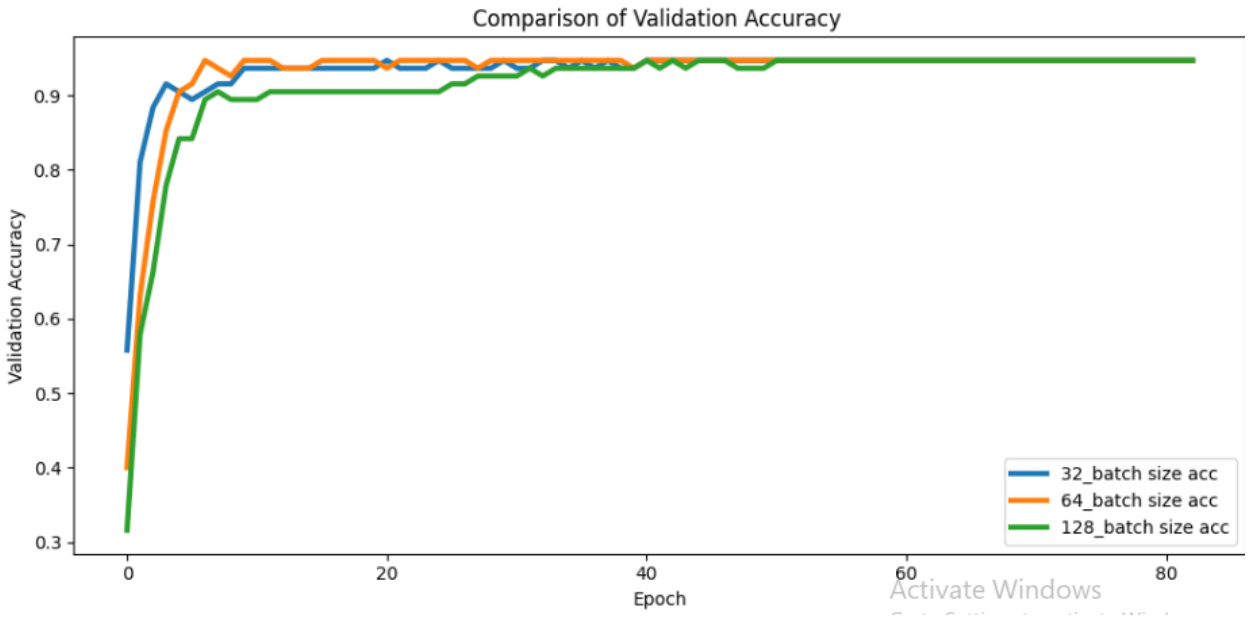
128 batch size achieved highest testing accuracy= 0.9843.

32 batch size testing accuracy= 0.9687.

64 batch size testing accuracy= 0.9531.







2.Hidden Size

For the one hidden layer we used different number of hidden nodes values (512, 256, 128).

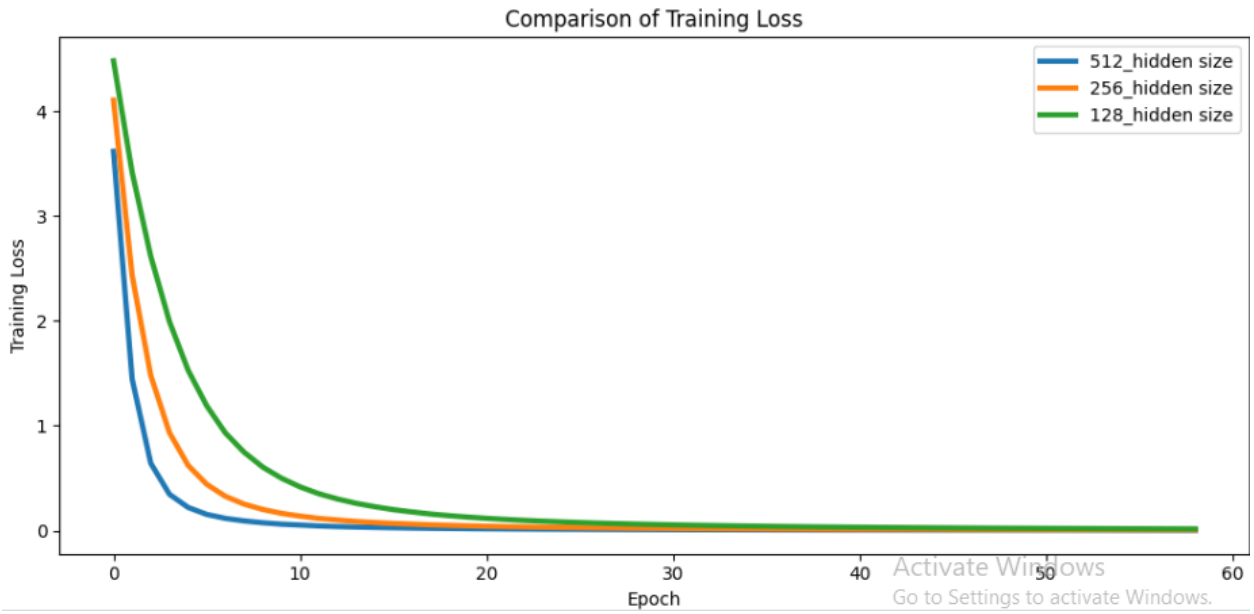
we get that:

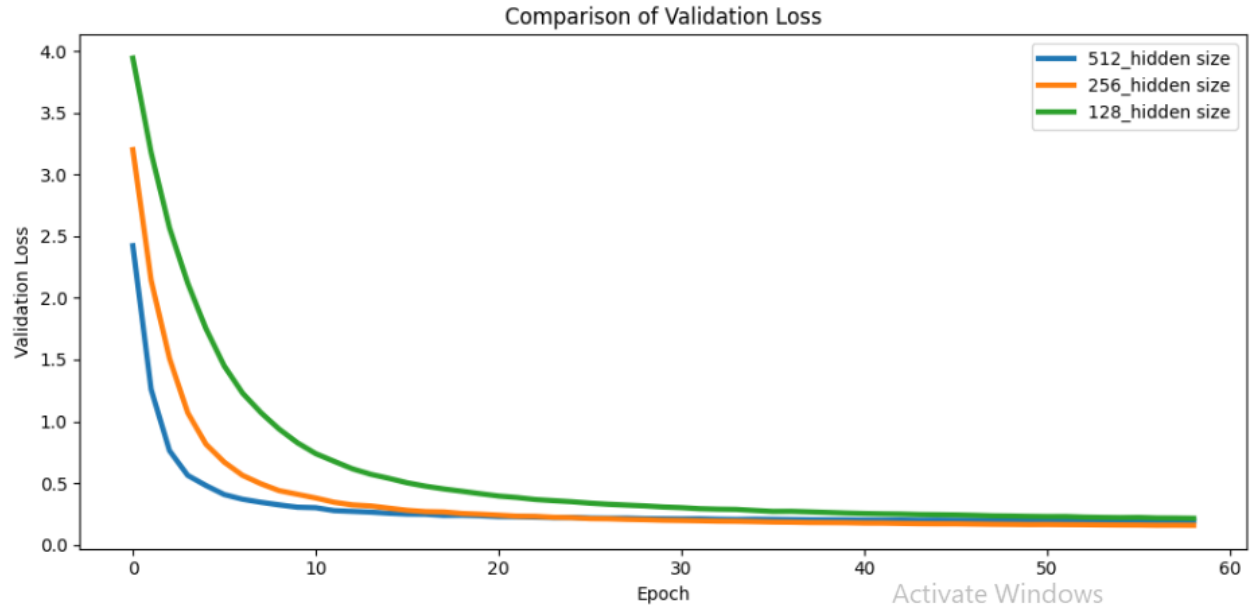
512 hidden nodes achieve highest accuracy and lowest loss very fast as shown in graphs below, but for long term work over the epochs 128 achieved higher accuracy.

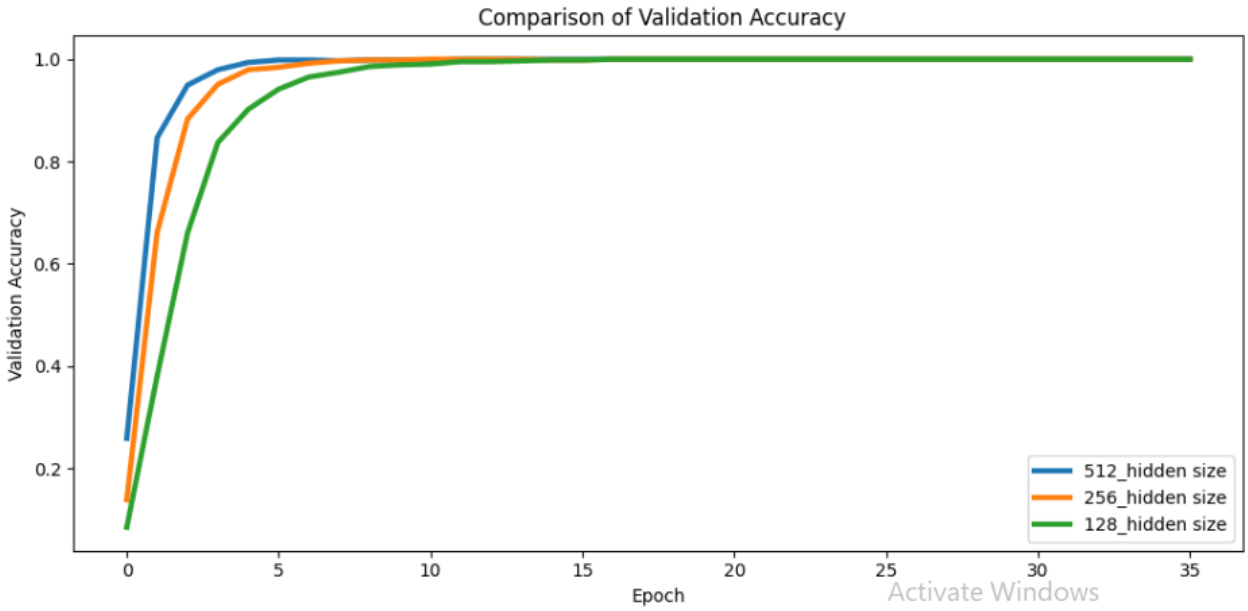
512 testing accuracy= 0.9375

256 testing accuracy= 0.9375

128 testing accuracy= 0.96875







3. Dropout

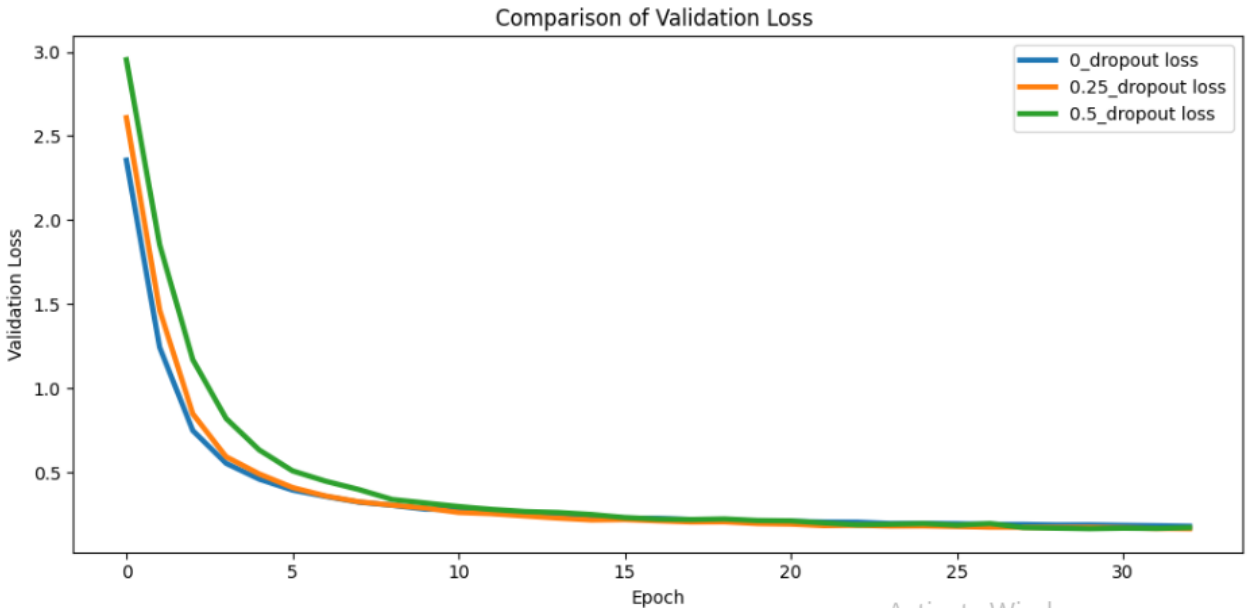
Trying different percentages (0, 0.25, 0.5) of dropout to determine the best and if it is helpful for our system or not.

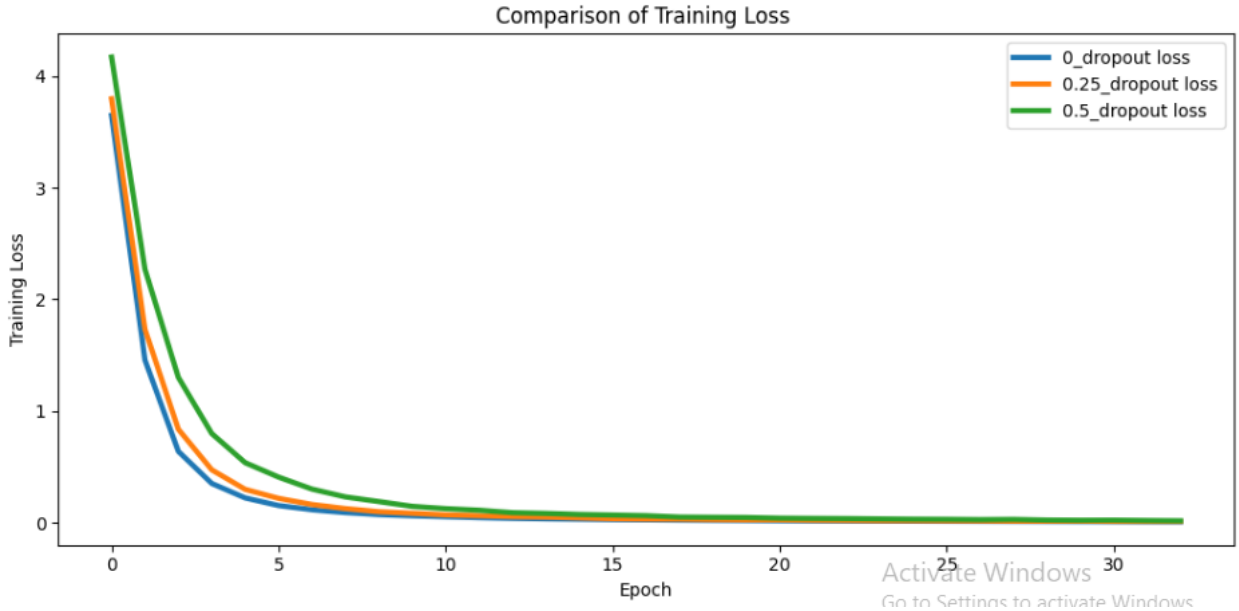
we get that: 0 drop out is the best as its testing accuracy= 0.9531

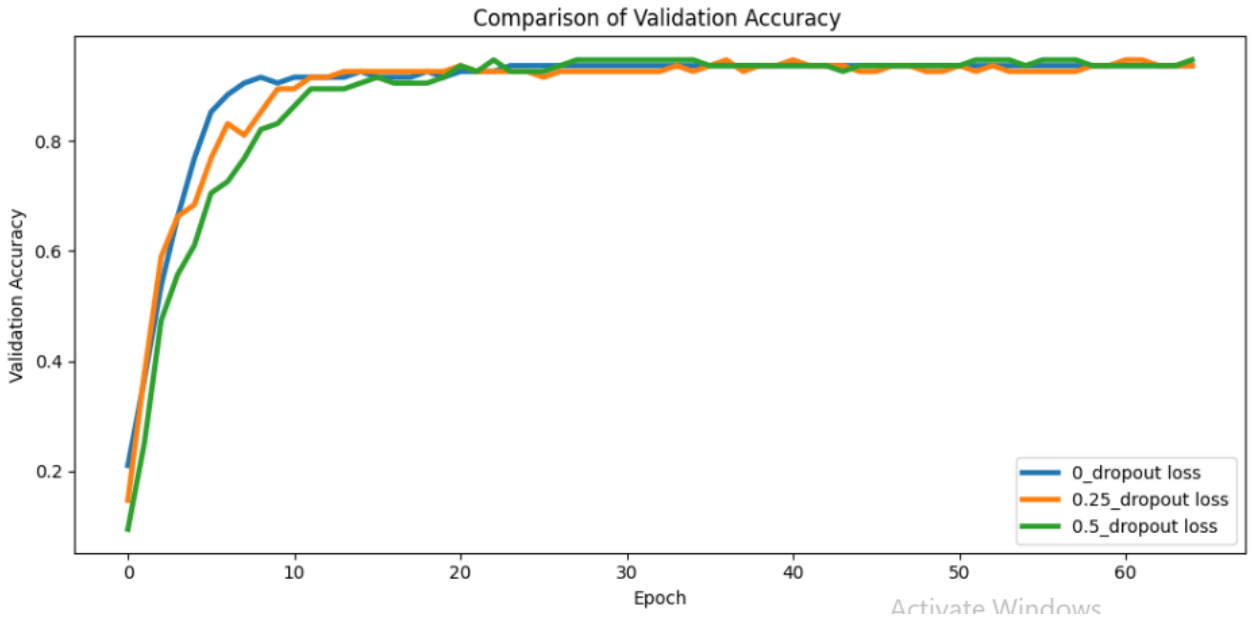
0.25 testing accuracy= 0.9375

0.5 testing accuracy= 0.9218

The graphs below show their performance in training and validation.







4.Optimizer

Comparing between SGD, Adam, RMSProp optimizers

we get that:

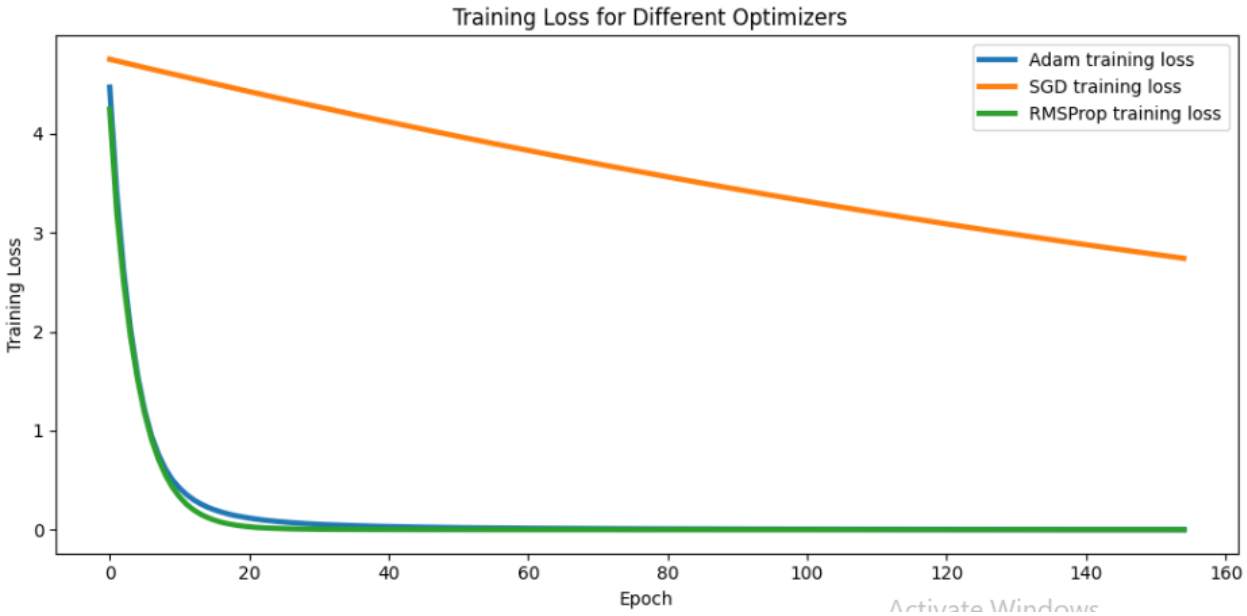
Adam optimizer is the best optimizer as:

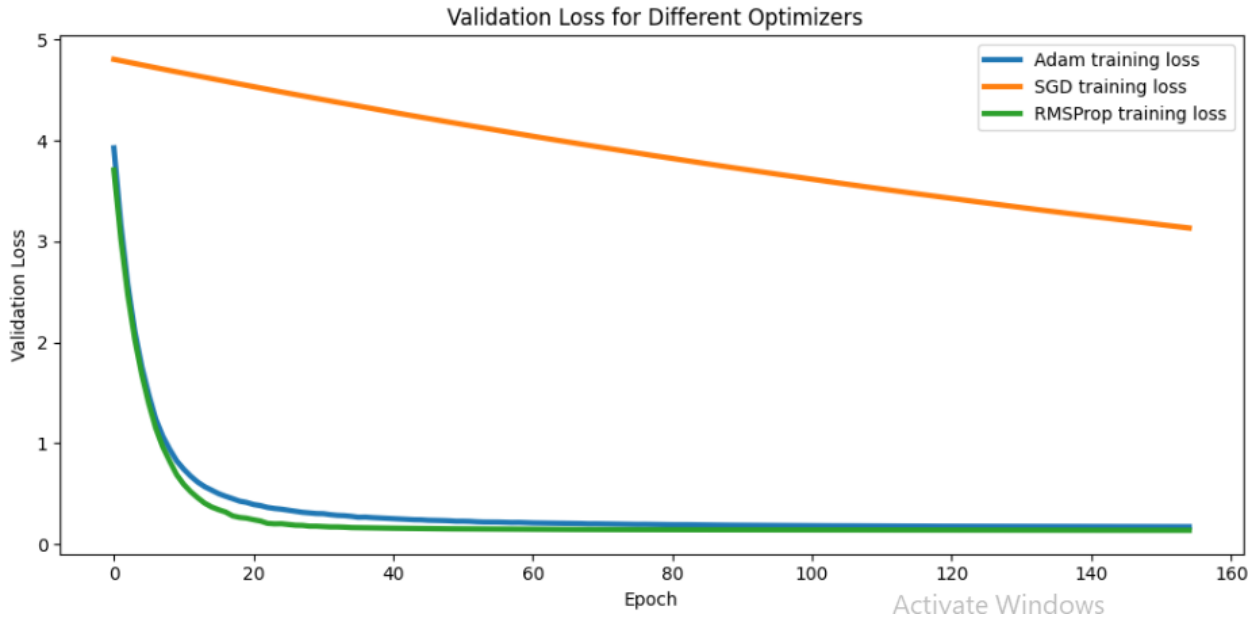
Adam testing accuracy= 0.9687

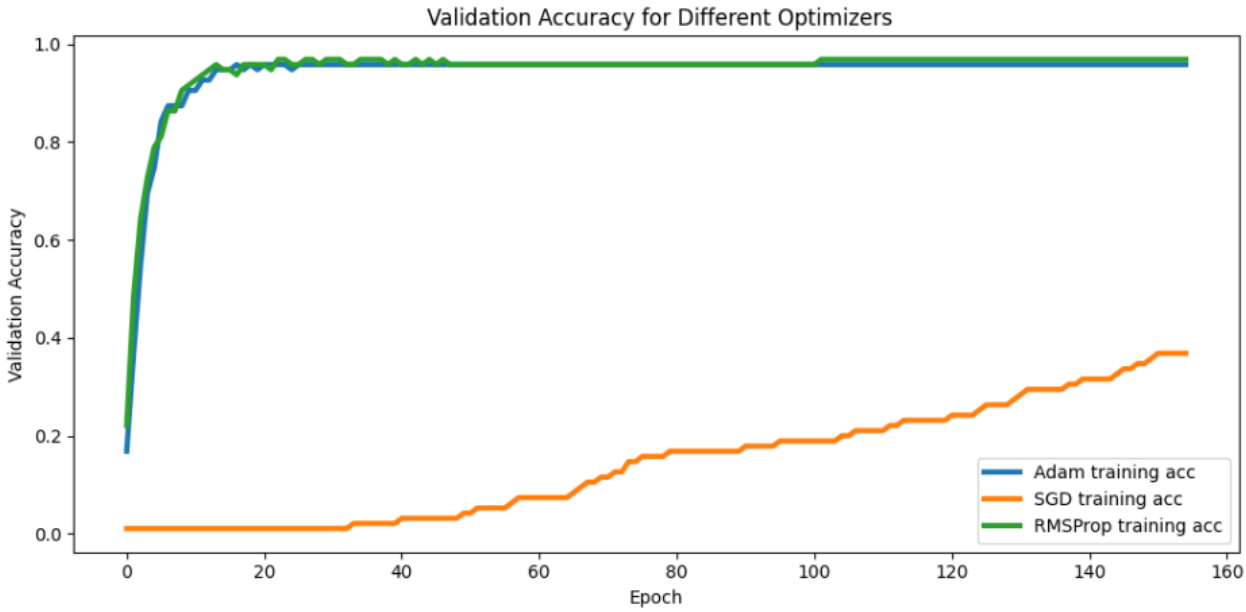
SGD testing accuracy= 0.7187

RMSprop testing accuracy= 0.9375.

performance as graphs below.


Training Loss for Different Optimizers
Adam training loss
SGD training loss
RMSProp training loss
Training Loss
Epoch
0
1
2
3
4
0
20
40
60
80
100
120
140
160


Validation Loss for Different Optimizers
Adam training loss
SGD training loss
RMSProp training loss
Validation Loss
Epoch
0
1
2
3
4
5
0
20
40
60
80
100
120
140
160



5.Regularization

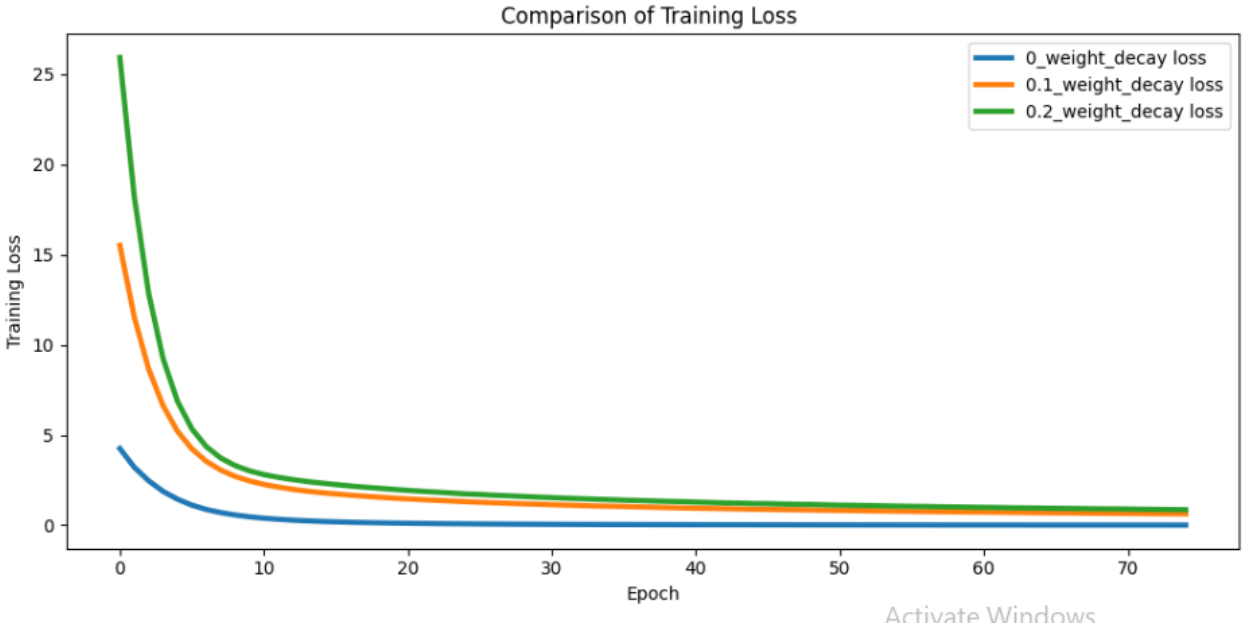
trying different regularization (weight_decay) values (0, 1, 2)

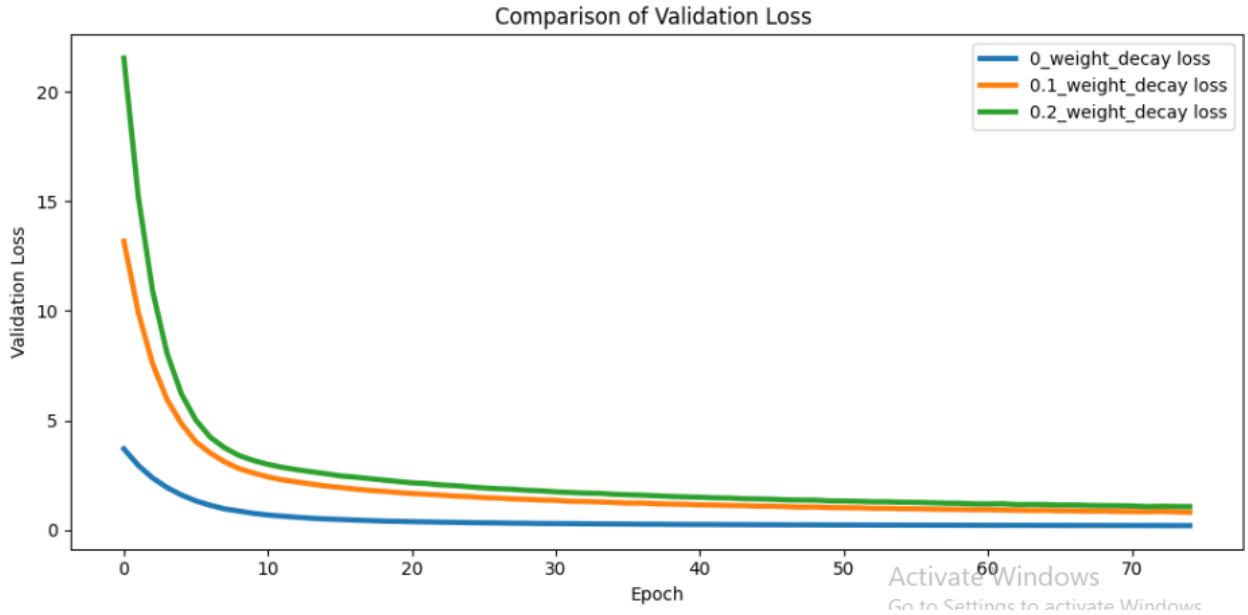
we get that:

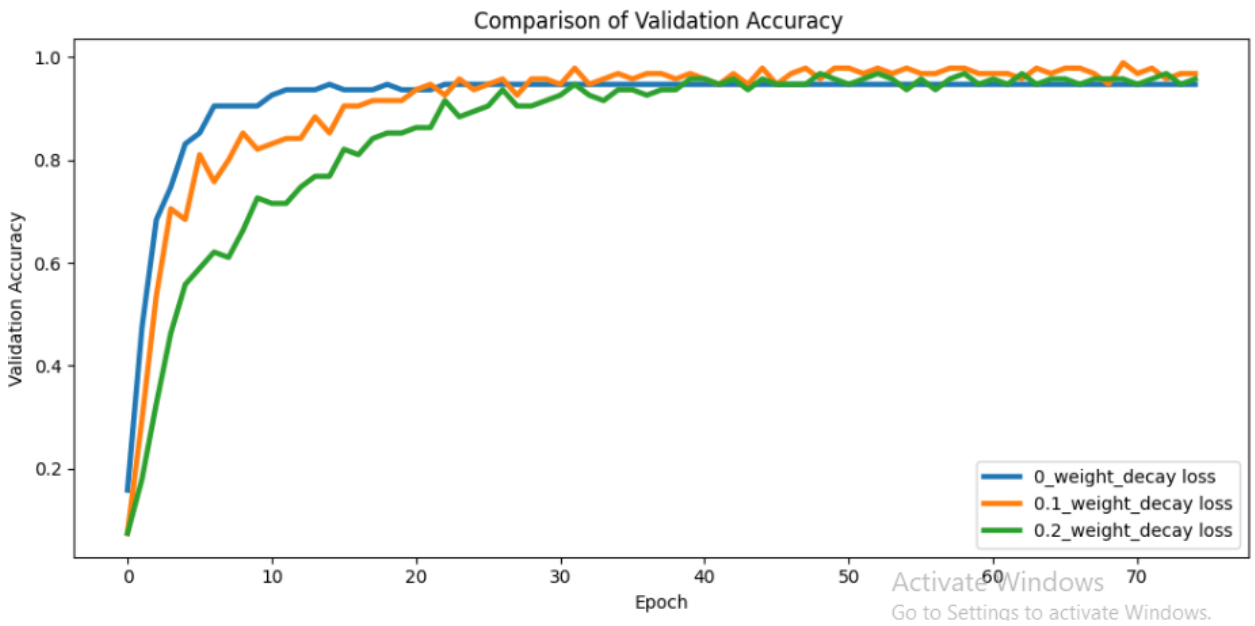
0 and .1 gives same testing accuracy =0.9531 but 0 has lower loss as in graphs

.2 testing accuracy= 0.9218.

0 is the best as it achieves low loss and more stable accuracy than the two other values.







6.Learning Rate

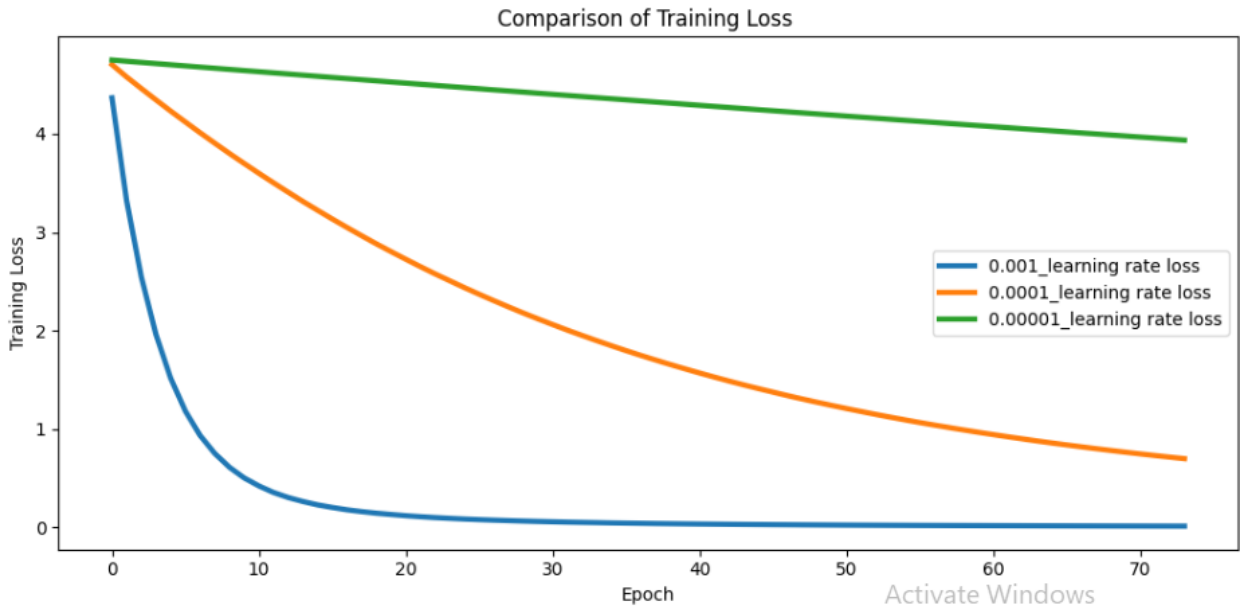
trying different values of our optimizer learning rate (0.001, 0.0001, 0.00001)

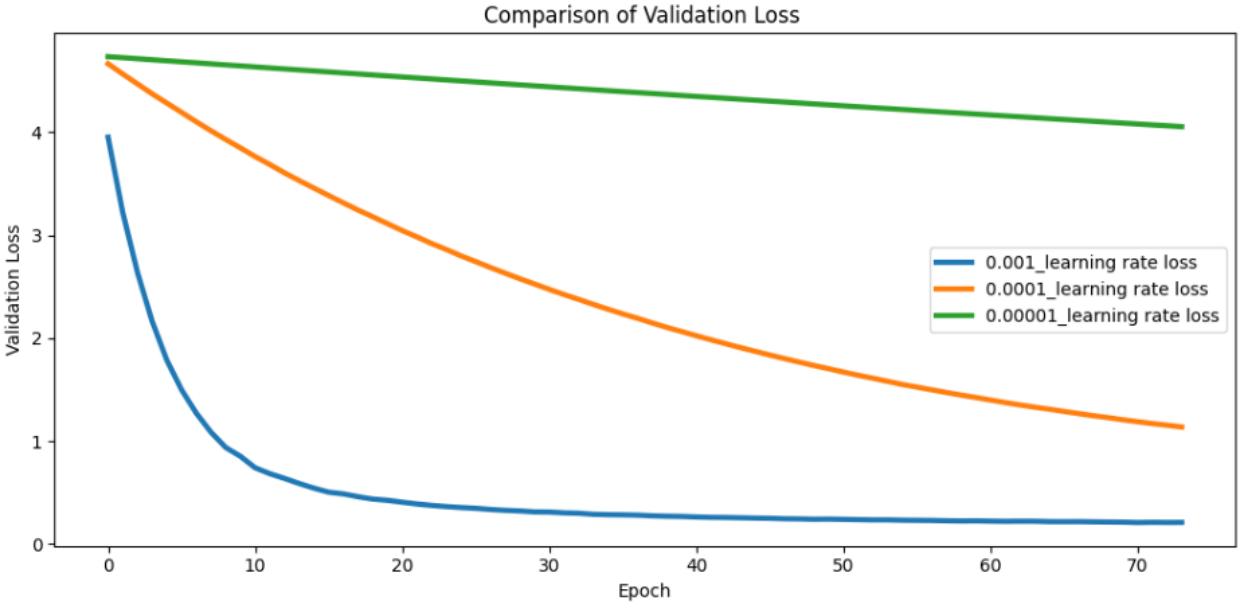
we get that:

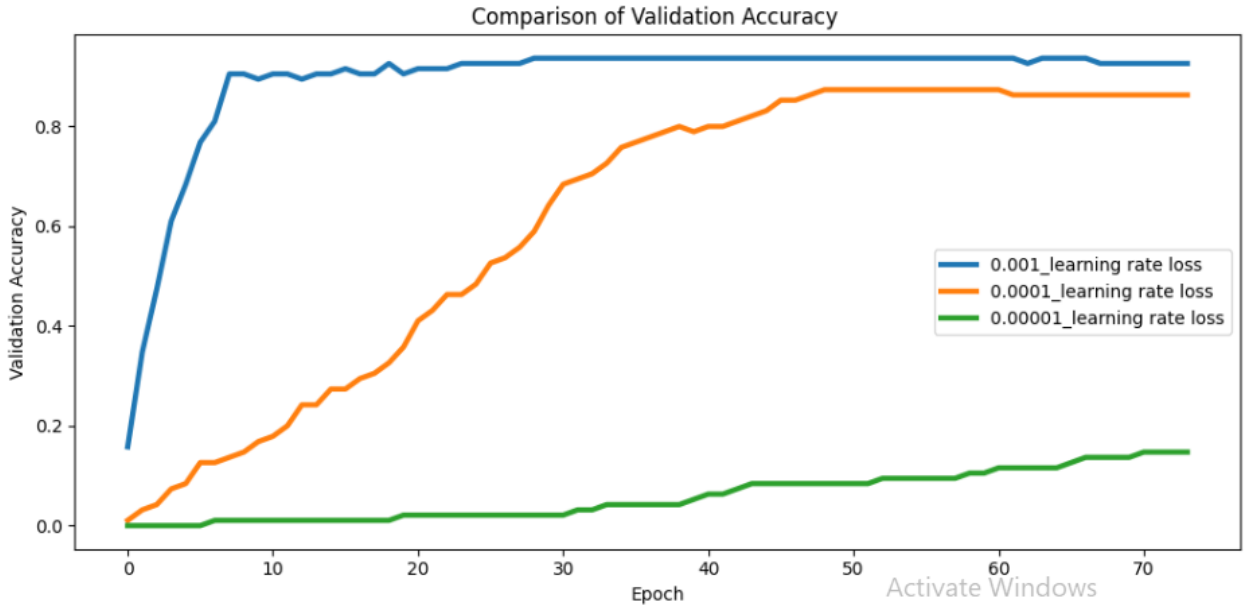
0.001 and 0.0001 gives same testing accuracy= 0.9531

0.00001 testing accuracy= 0.6875.

But 0.001 achieves the best performance as graphs.







Conclusion

We have effectively developed a deep learning model in this project to categorize various leaf species according to their characteristics. The most effective hyperparameters for model performance are Optimizer, Batch size, and Hidden size. As changing optimizer changes the whole performance of the model as it is clear in difference between Adam and SGD. Batch size increasing at first seems bad but after a range it makes opposite effect and increases the performance. Hidden size is so critical as increasing to large number may lead to overfitting, so as possible as can it is better to get small hidden size which in same time is suitable for our model. Finally, hyperparameter adjustments can be made to further enhance the model's performance.