Deep Learning and Neural Network

Leaf Classification

Name	NetID
Nourhan Abdelkerim	21nmma1
Sondos Mahmoud	21smam1
Farah Saber	21fsae

<u>Project description:</u>

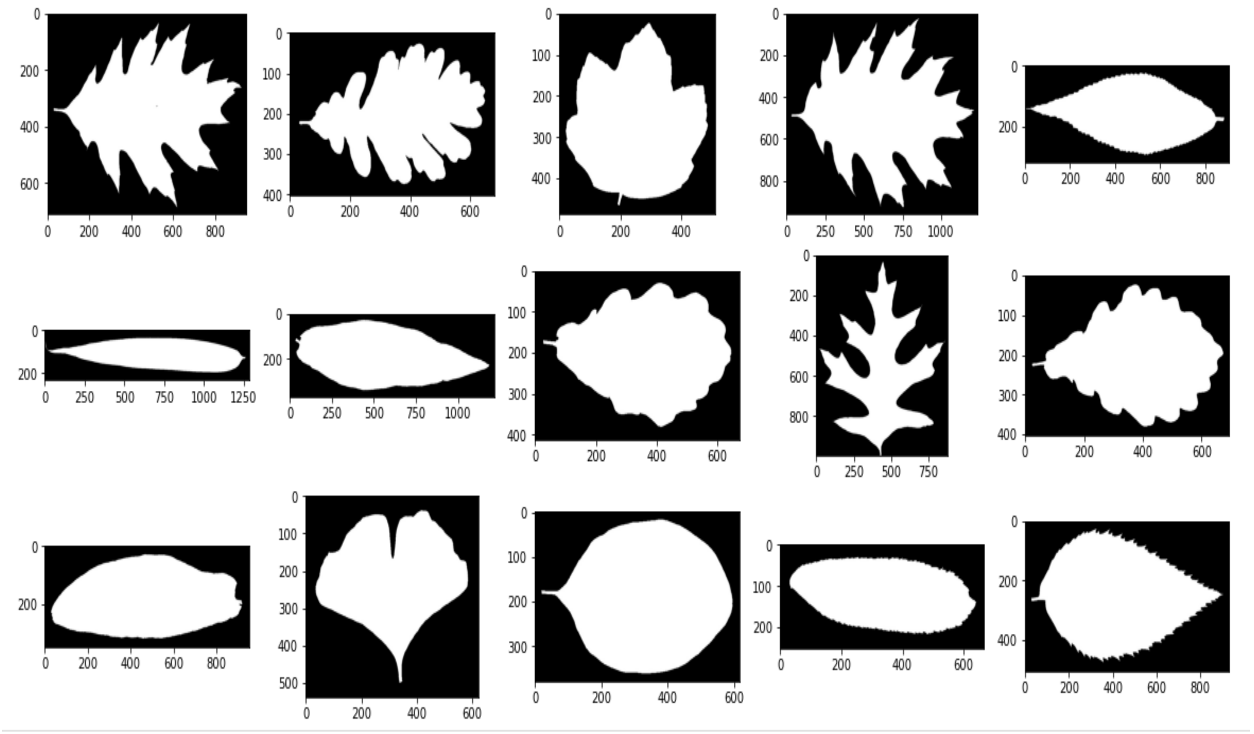
The objective of this project is to use extracted features, including shape, margin & texture, to accurately identify 99 species of plants. Leaves, due to their volume, prevalence, and unique characteristics, are an effective means of differentiating plant species.

In this project, 3-layer MLP model is implemented which consists of one input layer, one hidden layer with tanh activation and one output layer with SoftMax as activation function. This neural network will be used to classify the data in our dataset using TensorFlow and the built-in modules in keras to build the model.

The project includes many trials with different hyper parameters–, optimizers, learning rate and number of neurons in each layer to be able to reach the optimal solution that will best classify our data.

<u>Dataset:</u>

The dataset consists approximately 1,584 images of leaf specimens (16 samples each of 99 species) which have been converted to binary black leaves against white backgrounds. Three sets of features are also provided per image: a shape contiguous descriptor, an interior texture histogram, and a fine-scale margin histogram. For each feature, a 64-attribute vector is given per leaf sample.



Data fields:

1. id - an anonymous id unique to an image.
2. margin_1, margin_2, margin_3, ..., margin_64 - each of the 64 attribute vectors for the margin feature.
3. shape_1, shape_2, shape_3, ..., shape_64 - each of the 64 attribute vectors for the shape feature.
4. texture_1, texture_2, texture_3, ..., texture_64 - each of the 64 attribute vectors for the texture feature.

Methods:

Firstly, we check if there are missing values, duplicate values then we calculate z-score to check if we need to make standard scaler. Based on correlation between columns we drop useless columns.

After that, we made function contains the whole model with all hyperparameters and the evaluation of the model. This function contains the following hyperparameters:

1. Batch size
2. Hidden nodes
3. Learning rate
4. Epochs
5. L2-regularization
6. Dropout

We set the default values of hyperparameters on each optimizer then we set different values of hyperparameters. Also, we used a learning rate scheduler to change the learning rate for each epoch. Then we plot a graph for each model to represent the difference between training loss and validation loss, training accuracy and validation accuracy.

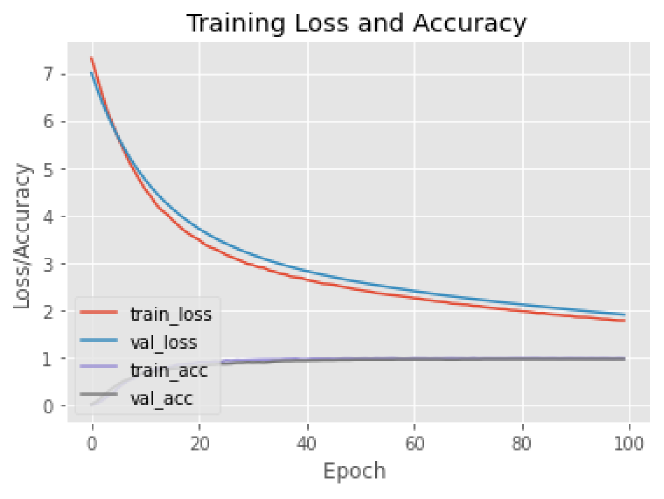
Optimizers:

1. SGD
2. Adam
3. RMSProp

- **SGD:**

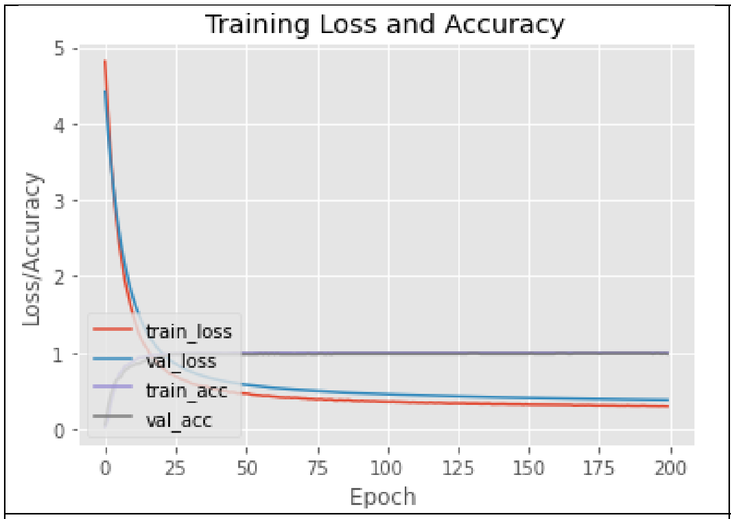
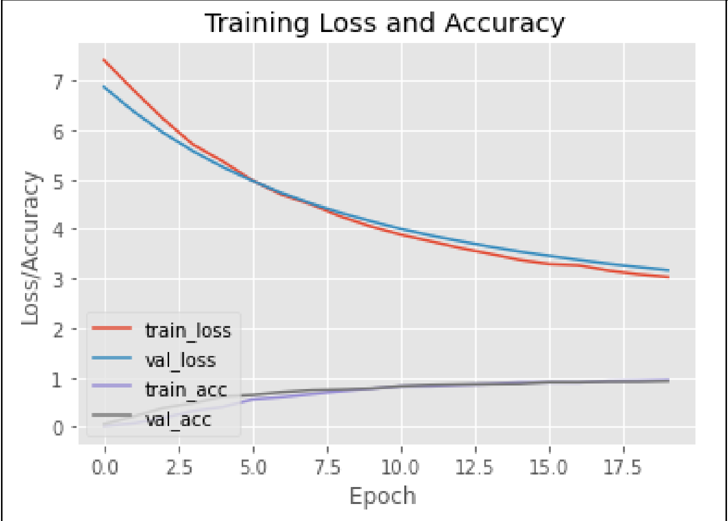
Stochastic gradient descent is considered an iterative method for optimizing an objective function with suitable smoothness properties. It can be regarded as a stochastic approximation of gradient descent optimization, since it replaces the actual gradient that is calculated from the entire data set by an estimate thereof calculated from a randomly selected subset of the data. This reduces the computational complexity especially in high-dimensional optimization problems which help in achieving faster iterations in trade for a lower convergence rate. SGD was able to solve the Gradient Descent problem by using only single records to updates parameters. But still SGD is slow to converge as it needs forward and backward propagation for each record and therefore the path to reach global minima becomes very noisy.

Optimizer (SGD)	Hidden nodes	L2-regularization	Learning rate	Dropout	epochs	Accuracy
First Trial	800	0.01	0.0001	0.4	100	0.9747
Second Trial	1070	0.001	0.01	0.4	200	0.984848
Third Trial	1000	0.01	0.1	0.6	20	0.9242



First Trial

loss: 1.9129 - accuracy: 0.97477

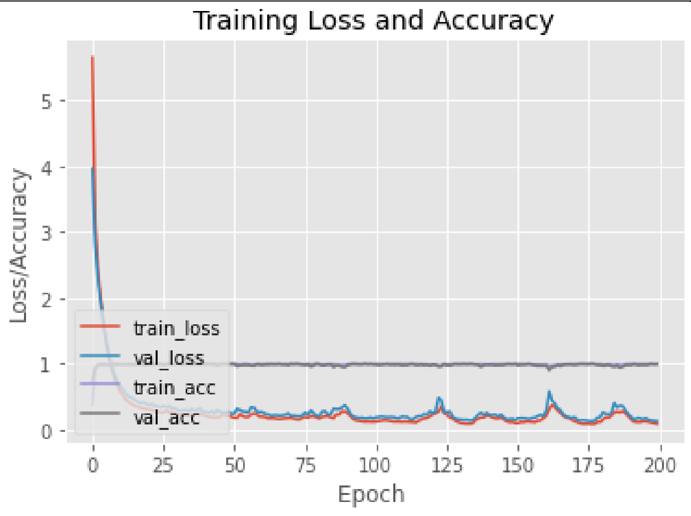
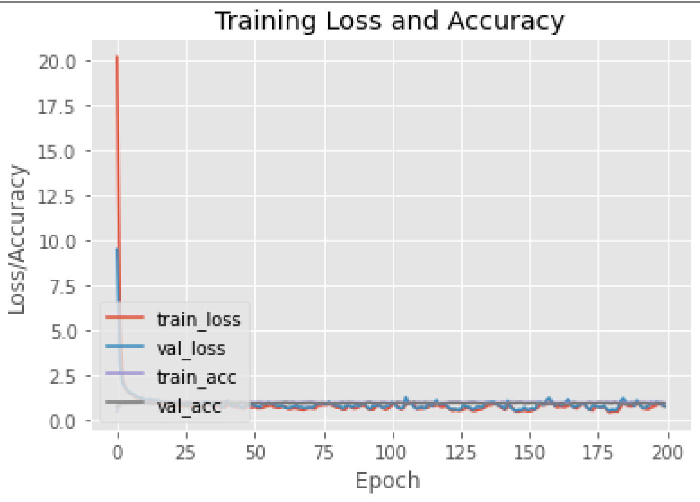
	**Second Trial** loss: 0.379946 - accuracy: 0.984848
	**Third Trial** loss: 3.1674 - accuracy: 0.924242

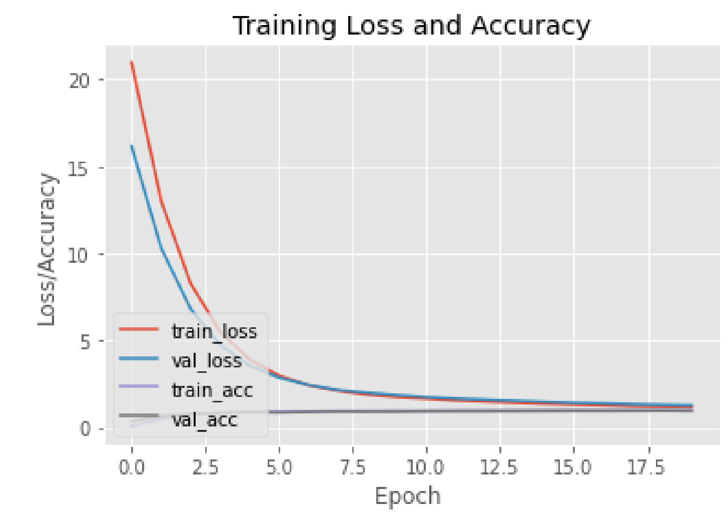
1. As we show, when we decrease the learning rate, L2-regularization with epochs=200 mean that the overfitting will also decrease with highest accuracy and low loss.
2. When we decrease the learning rate, L2-regularization with epochs=100 mean that the overfitting will also decrease but the accuracy is smaller than when epochs=200.
3. When we increase the dropout mean that overfitting will decrease.

- **Adam:**

Adam optimization extends on stochastic gradient descent to solve non-convex problems faster that is based on adaptive estimation of first order and second-order moments. The method is efficient with working on large problem involving a lot of data or parameters. It requires less memory and efficient.

Optimizer (Adam)	Hidden nodes	L2-regularization	Learning rate	Dropout	epochs	Accuracy
First Trial	1092	0.01	0.001	0.4	200	0.9949
Second Trial	300	0.1	0.01	0.4	20	0.9697
Third Trial	3000	0.1	0.1	0.4	200	0.9192

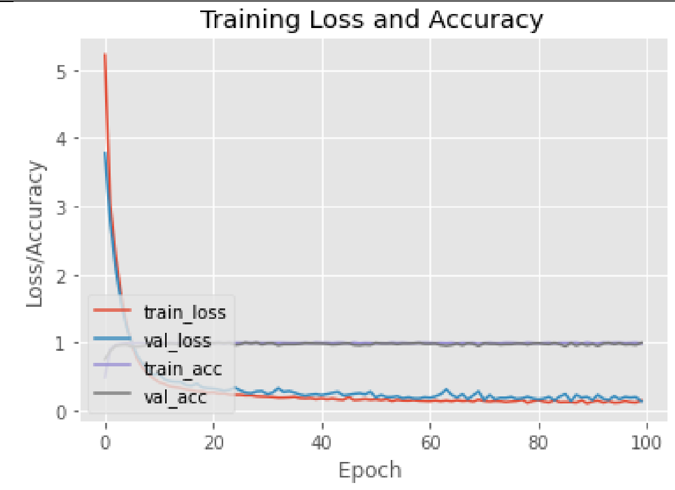
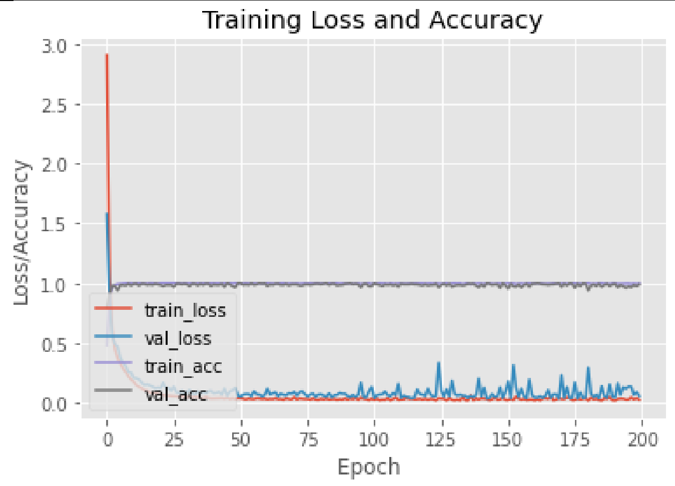
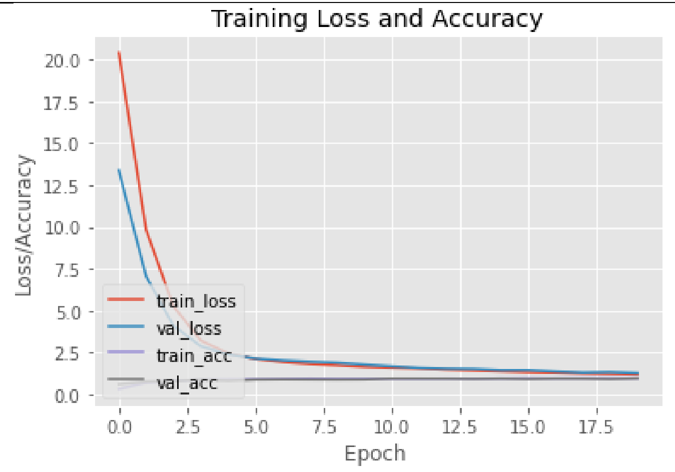
Graph	Accuracy
	First Trial: Loss:0.1282 Accuracy: 0.99494951
	Second Trial: Loss:1.283 Accuracy: 0.9696

	Third Trial: Loss:0.7734 Accuracy: 0.91919

- **<u>RMSProp:</u>**

The RMSprop optimizer is like the gradient descent algorithm with momentum. The RMSprop optimizer restricts the oscillations in the vertical direction. As a result, we can increase our learning rate and our algorithm could take larger steps converging faster. The difference between RMSprop and gradient descent is on how the gradients are calculated.

Optimizer (RMSProp)	Hidden nodes	L2-regularization	Learning rate	Dropout	epochs	Accuracy
First Trial	1092	0.01	0.1	0.4	100	0.9949
Second Trial	1070	0.001	0.01	0.4	200	0.98989
Third Trial	1000	0.1	0.001	0.6	20	0.9596

Graph	Accuracy
	First Trial: Loss:0.1349 Accuracy:0.9949
	Second Trial: Loss:0.0572 Accuracy:0.98989
	Third Trial: Loss:1.2777 Accuracy: 0.9596

Now we will display results with Learning Rate Schedule

Optimizers	Hidden nodes	L2-regularization	Dropout	Epochs	Loss-Accuracy
Adam	1092	0.0001	0.4	100	0.0876 - 0.9899
RMSProp	900	0.01	0.4	100	0.2836-0.9949
SGD	1092	0.01	0.4	20	3.44-0.9040

Note:

RMSProp gives us the highest accuracy but Adam gives us the least loss with high accuracy

Conclusion

To prevent the overfitting, we use dropout and L2-regularization. According to the above outputs, the Adam & and RMSProp optimizers are close in accuracy. However, Adam is the optimization to choose the least loss function with highest accuracy.