CSE463
Neural Networks

حازم اشرف مندي	1700420
ثريا احمد عبد الظاهر عبد العزيز	1700391
مريم هاني فوزي عوض	1701410

Introduction

Binary Classification is when you are trying to classify an input as one of two outcomes, for instance, do you think a student will pass or fail a class?. Does this photo belong to a dog or a cat?.

However, real-world problems aren't simply a yes or no question. This brings us to the idea of multiclass classification, in which inputs can be classified as one of many outputs. like you need to predict an image of a digit between 0–9.

Problem definition and importance

Problems in dataset definition

CIFAR-100 is a labeled subset of 80 million tiny images dataset where CIFAR stands for Canadian Institute For Advanced Research. The images were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The dataset consists of 60000 colored images (50000 training and 10000 test) of 32 × 32 pixels in 100 classes grouped into 20 superclasses. Each image has a fine label (class) and a coarse label (superclass). The Python version of this dataset has been downloaded from the website of the University of Toronto Computer Science and used for the project

Shallow neural network :

shallow neural networks is a term used to describe NN that usually has only one hidden layer as opposed to deep NN which has several hidden layers, often of various types of architectures.

The first problem is the shallow neural network complexity. So we do feature engineering extraction, like how to benefit from our features in the most efficient way to get similar results without having to deal with all the image pixels as inputs. As the more input nodes the higher the complexity.

Our problem is how to design one layer to be able to classify an image from data set, how to choose a good optimizer as it has a big effect on our results and number of nodes in the layer,

Deep neural network:

Deep neural networks with the right architecture achieve better results than shallow ones with the same computational power.

However, there are many problems we have to deal with; like how to choose a good optimizer for our case study, network structure, how to reduce the algorithm complexity, how to increase the accuracy

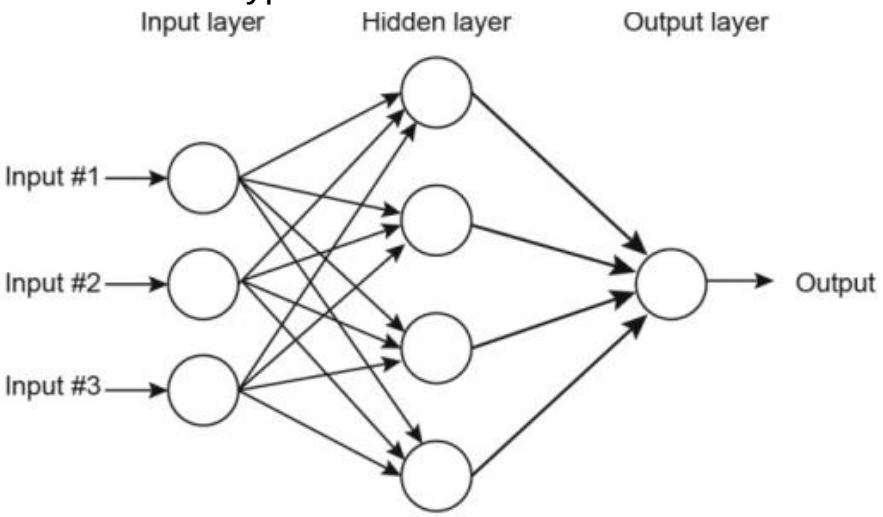
Network architecture : it plays an important role in our problem, it has a big effect on our results. As the number of layers, number of nodes at each layer, different techniques ..etc

Optimizers: there are many optimizers. Every optimizer has its features. We try to find a quick and accurate optimizer.

Complexity: to achieve good accuracy you have to train your algorithm with a good number of epochs, one epoch recording to the algorithm takes many seconds which may exceed one minute, so to check our results it takes a few hours.

Methods and Algorithms
1- Shallow neural network

Support vector machine (SVM) algorithm is a very powerful algorithm for classification problems. The original usage of SVM was only in binary classification. Subsequently, several variants of multiclass SVMs have been proposed; one of them was Weston and Watkins[2].Weston Watkins algorithm is efficient and accurate compared to the other mulit-calss support vector machine algorithm.
So we took the Weston and Watkins SVM approach.
This is what a typical shallow neural network looks like



For the input layer we tried first to take all the pixels of the image sample 3072 nodes, but the model complexity was very high took approximately 4 hours to complete one epoch.So we've done some manual feature engineering by removing two of the colors channels as one color was clear for a human to classify reducing the input dimension to 1024 nodes, unfortunately it was complex also.So we used the Principal Component Analysis algorithm to reduce the dimensionality of the problem to

1. 99 features, but the model was very complex as well.
2. 10 features, which time to complete one epoch was acceptable.
3. 5 features and 2 features, which didn't make as much difference in performance as the second option.

We chose to continue with the second option.

For initialization we initialized the weights with normal distribution and normalized the data.
Prediction rule argmax$_t$ $(Wt^T\ Xi)$

For the hidden layer loss function :
d_t = max $(1 + Wt^T\ Xi - Wyi^T\ Xi, 0)$, r ∈ { 1,2, … , (k-1) − {yi-1 }}
$L_i = \sum_{t=0, t \neq Yi-1}^{k-1} d_t$
Update rule: δ_{ji} = 1 if 1 + $\boldsymbol{WjXi} - \boldsymbol{Wyi}\ \boldsymbol{Xi} > 0$ δ_{ji} = 0 otherwise

Cost function : sum of the batch losses. Regularisation function is L2.

For optimization techniques implemented :

- Vanilla Gradient descent
- Adam
- Mini batch gradient descent
- Mini batch Adam

The mini batch adam optimization algorithm has been used in our model for optimization as it converges faster and so was more computationally efficient than other optimization algorithms. And less memory usage.
The batch size is 100 for managing memory resources, and alpha, beta1=0.9 ,and beta 2= 0.999
Through validation against the validation set

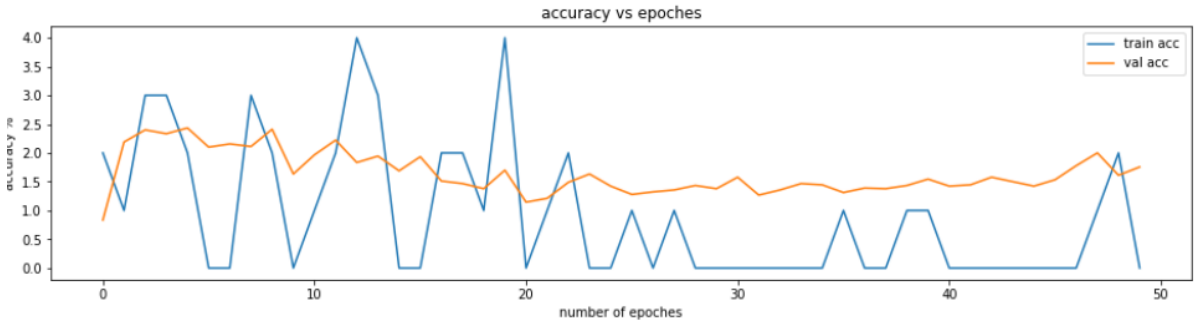


Fig 1.1 train-validation accuracy with alpha = 0.1

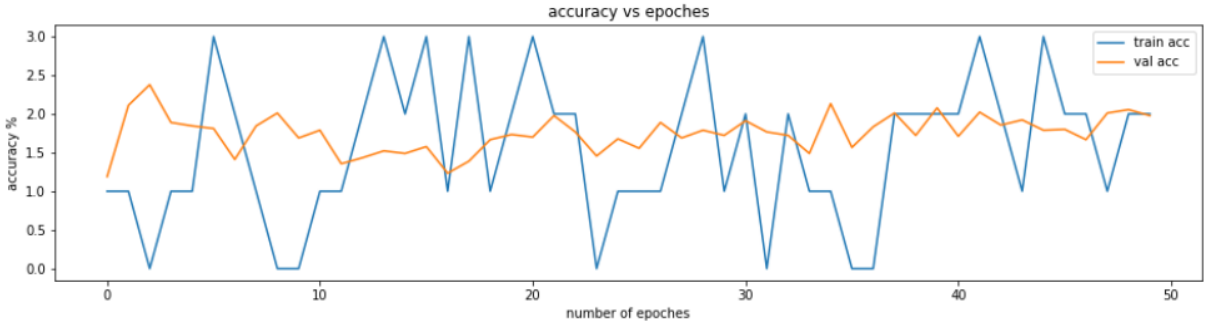


Fig 1.2 train-validation accuracy with alpha = 0.01

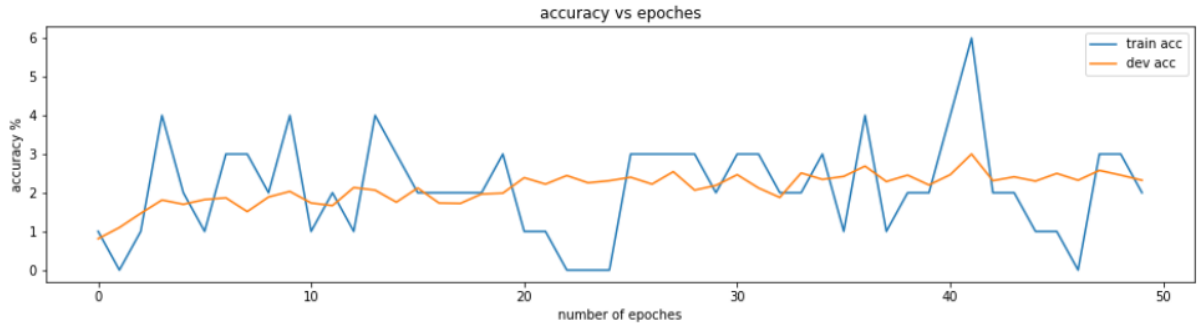


Fig 1.3 train-validation accuracy with alpha = 0.001

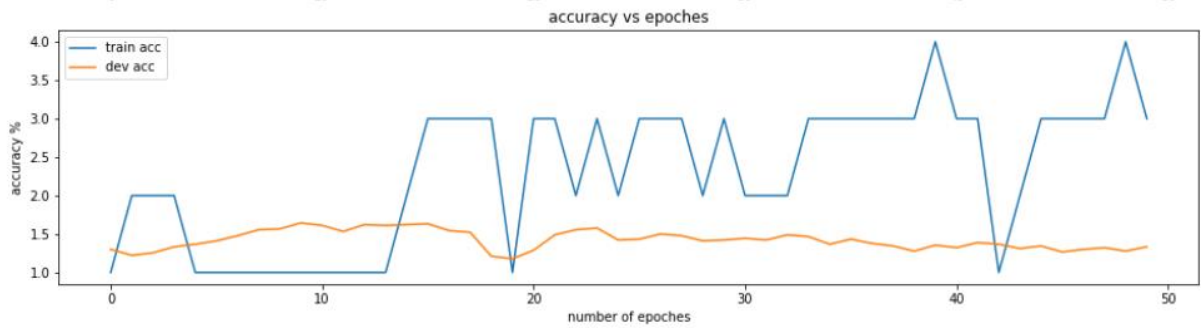


Fig 1.4 train-validation accuracy with alpha = 0.0001

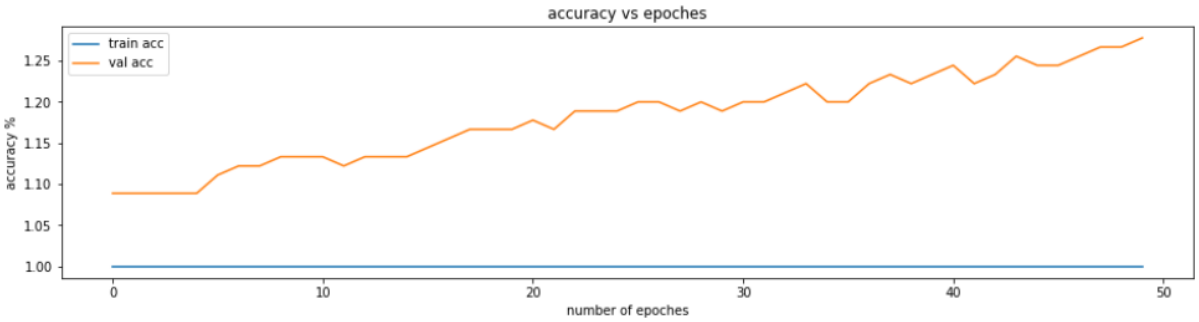


Fig 1.5 train-validation accuracy with alpha = 0.00001

We chose alpha = 0.0001

2- Deep neural network

a) categorical_crossentropy: Used as a loss function for multi-class classification model where there are two or more output labels. The output label is assigned one-hot category encoding value in form of 0s and 1. The output label, if present in integer form, is converted into categorical encoding using Keras.

b)

we’ve implemented three blocks of [CONV_1 -> Elu activation -> CONV _2-> Elu activation -> max pooling ->Dropout] and the last block [flatten -> dense 0> Elu -> SoftMax].

Conv_1 uses padding of type same to ensure that the dimension of output is the same as the dimension of input.

Conv_2 doesn’t use padding for dimensionality reduction

We used Elu activation as it has better generalization performance than Relu, fully continuous, and differentiable function.

Max pooling to reduce the dimensionality of the last layer to have as the pooling size is (2,2).

Dropout to get a better generalization of the model to the testing data. Lastly, the SoftMax is just for prediction.

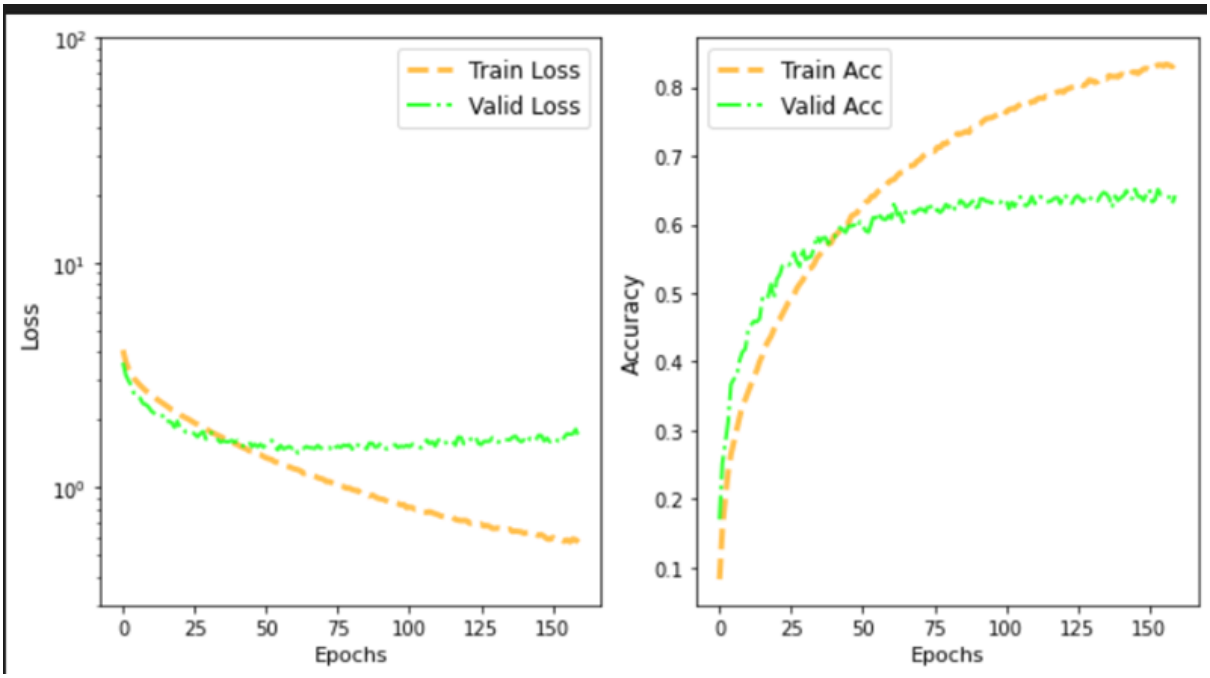
Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 128)	3584
activation (Activation)	(None, 32, 32, 128)	0
conv2d_1 (Conv2D)	(None, 30, 30, 128)	147584
activation_1 (Activation)	(None, 30, 30, 128)	0
max_pooling2d (MaxPooling2D)	(None, 15, 15, 128)	0
conv2d_2 (Conv2D)	(None, 15, 15, 256)	295168
activation_2 (Activation)	(None, 15, 15, 256)	0
conv2d_3 (Conv2D)	(None, 13, 13, 256)	590080
activation_3 (Activation)	(None, 13, 13, 256)	0
max_pooling2d_1 (MaxPooling2	(None, 6, 6, 256)	0
dropout (Dropout)	(None, 6, 6, 256)	0
conv2d_4 (Conv2D)	(None, 6, 6, 512)	1180160

activation_4 (Activation)	(None, 6, 6, 512)	0
conv2d_5 (Conv2D)	(None, 4, 4, 512)	2359808
activation_5 (Activation)	(None, 4, 4, 512)	0
max_pooling2d_2 (MaxPooling2	(None, 2, 2, 512)	0
dropout_1 (Dropout)	(None, 2, 2, 512)	0
flatten (Flatten)	(None, 2048)	0
dense (Dense)	(None, 1024)	2098176
activation_6 (Activation)	(None, 1024)	0
dropout_2 (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 100)	102500
activation_7 (Activation)	(None, 100)	0

==========

c)




```
Model Accuracy = 0.64
Actual Label = mountain vs. Predicted Label = train
Actual Label = forest vs. Predicted Label = forest
Actual Label = seal vs. Predicted Label = otter
Actual Label = mushroom vs. Predicted Label = mushroom
Actual Label = sea vs. Predicted Label = sea
Actual Label = tulip vs. Predicted Label = spider
Actual Label = camel vs. Predicted Label = bicycle
Actual Label = butterfly vs. Predicted Label = shrew
Actual Label = cloud vs. Predicted Label = cloud
Actual Label = apple vs. Predicted Label = apple
Actual Label = sea vs. Predicted Label = dinosaur
Actual Label = skunk vs. Predicted Label = skunk
Actual Label = streetcar vs. Predicted Label = streetcar
Actual Label = rocket vs. Predicted Label = rocket
Actual Label = lamp vs. Predicted Label = lamp
Actual Label = lion vs. Predicted Label = lion
Actual Label = tulip vs. Predicted Label = tulip
Actual Label = wolf vs. Predicted Label = wolf
Actual Label = rose vs. Predicted Label = rose
Actual Label = orange vs. Predicted Label = orange
Actual Label = rose vs. Predicted Label = rose
```

d) Optimization using RMSprop optimizer and initiate it with (learning_rate=0.0001, decay=1e-6).

Algorithms:

1) We loaded the data set by using: (x_train, y_train), (x_test, y_test) = cifar100.load_data()

2) Convert class vectors to binary class matrices by :

y_train = to_categorical(y_train, num_classes)

y_test = to_categorical(y_test, num_classes)

3) Normalizing by: x_train /= 255.

x_test /= 255.

4) Preprocessed training images by: tf.keras.preprocessing.image.ImageDataGenerator(zoom_range=0.2,

width_shift_range=0.1,

height_shift_range = 0.1,

horizontal_flip=True)

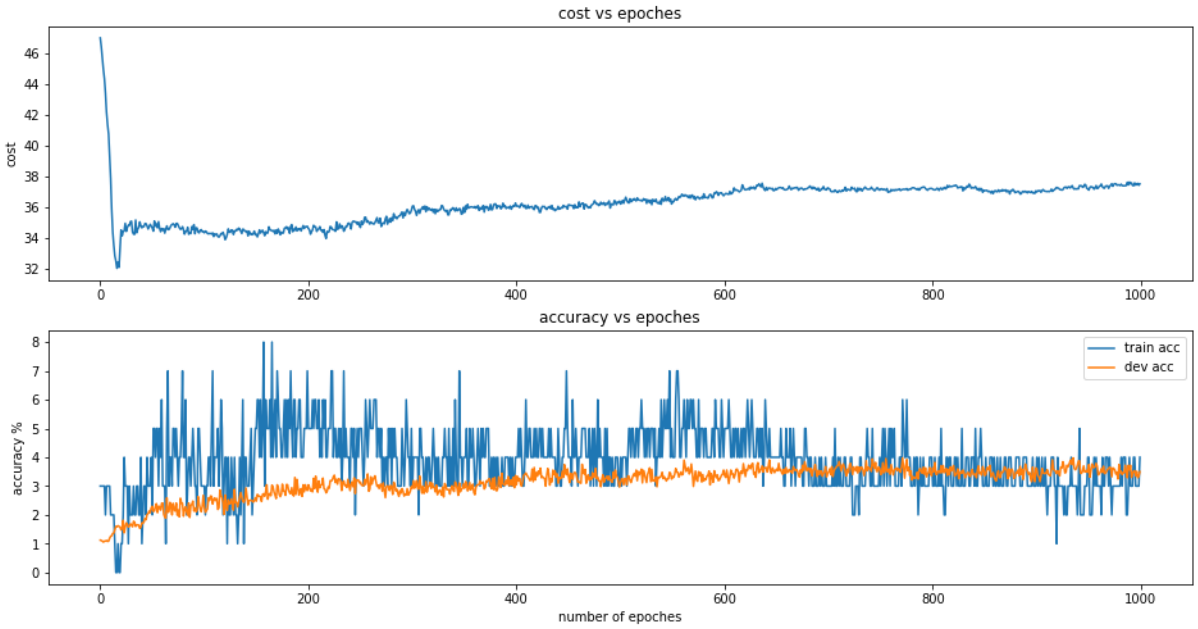
5) Preprocessed validation images by: tf.keras.preprocessing.image.ImageDataGenerator()

6) Creating deep neural network .

Experimental Results:

a) Shallow network

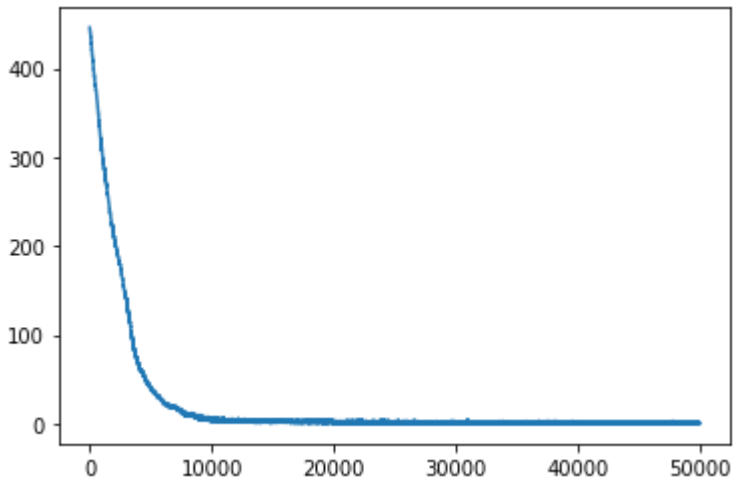
The First 1000 iteration were

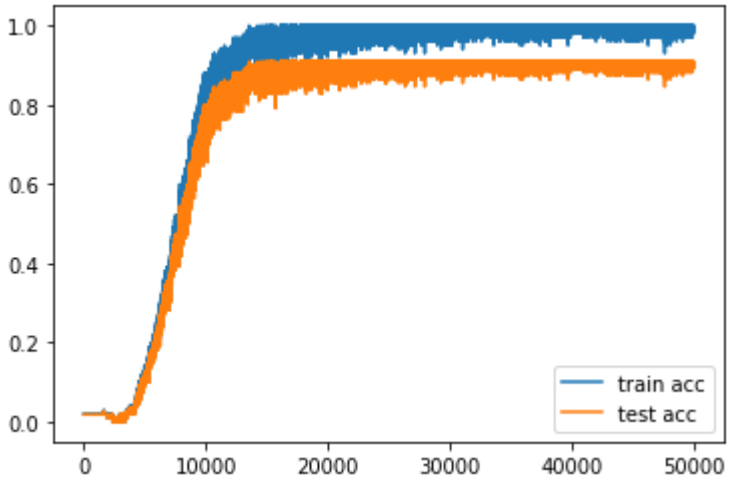


And the test accuracy = 3.033 %

We add regularisation term lamda = 0.2 but the model wasn't even near the overfitting and the test accuracy = 1% .

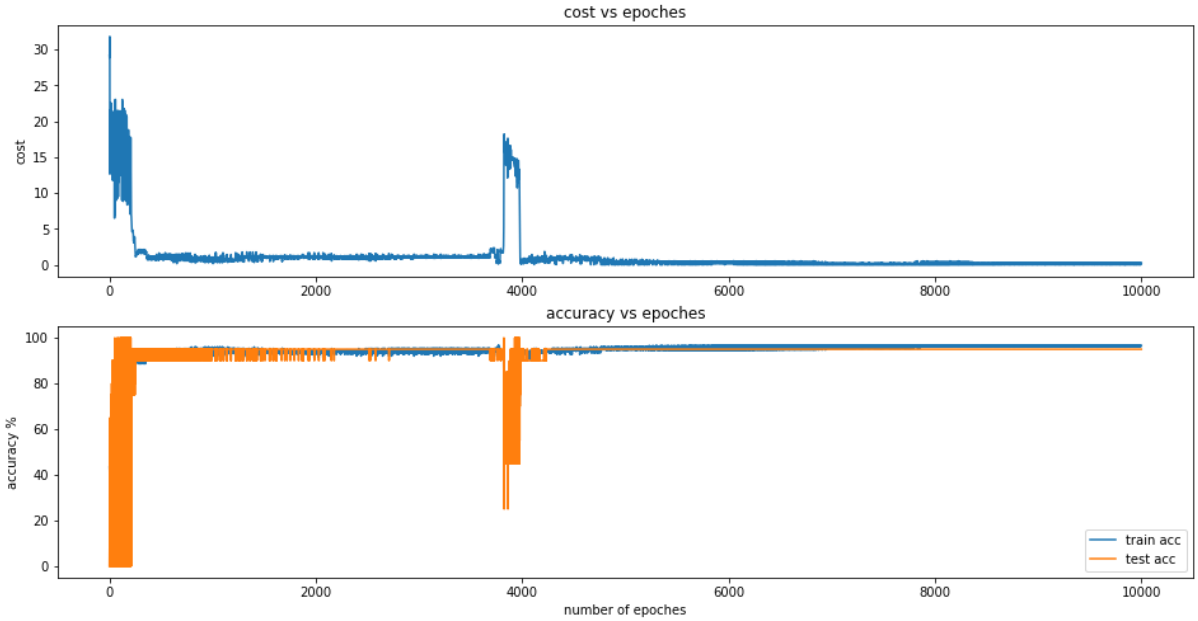
Also we tried to only train the model on 100 samples only it did take 70k epochs to reach 100% train accuracy and 89%test accuracy





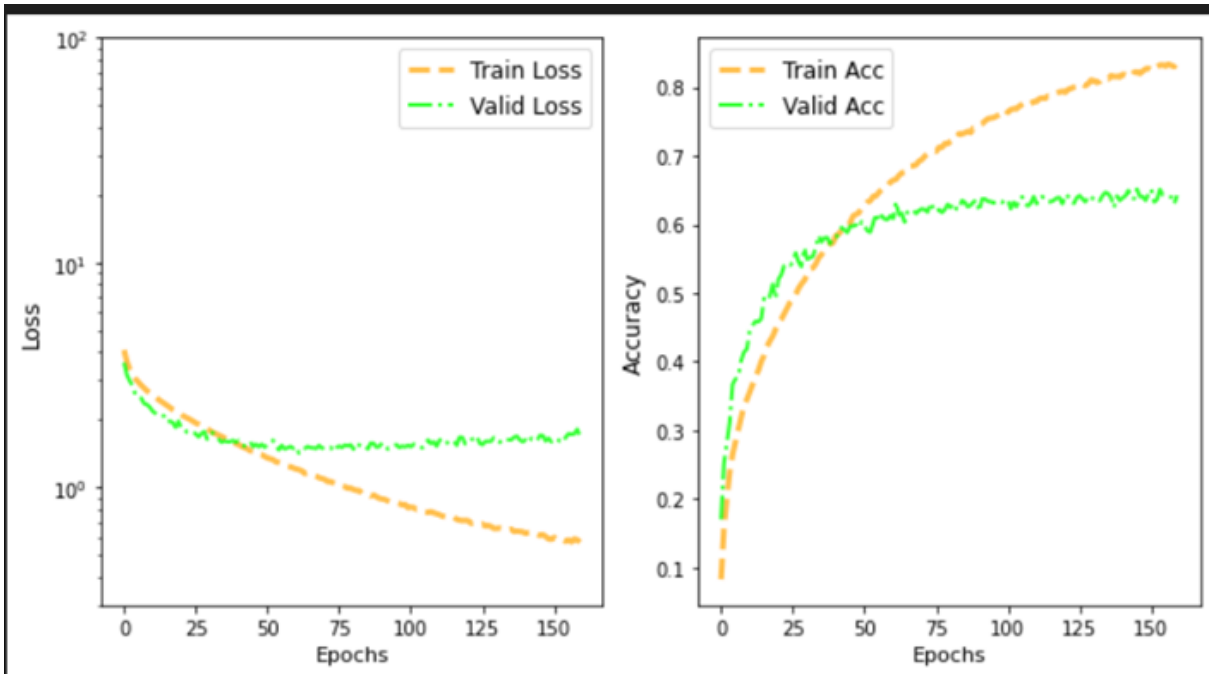
Which is very complex and not acceptable in terms of time and resources.

however, it does well on small and easy datasets like the Wheat-Seeds [link] got 97.67% on training and 82% on testing



Lastly, weston watkins mcsvm doesn’t perform well on today's problems compared to deep neural networks.

b) deep neural network




```
Model Accuracy = 0.64
Actual Label = mountain vs. Predicted Label = train
Actual Label = forest vs. Predicted Label = forest
Actual Label = seal vs. Predicted Label = otter
Actual Label = mushroom vs. Predicted Label = mushroom
Actual Label = sea vs. Predicted Label = sea
Actual Label = tulip vs. Predicted Label = spider
Actual Label = camel vs. Predicted Label = bicycle
Actual Label = butterfly vs. Predicted Label = shrew
Actual Label = cloud vs. Predicted Label = cloud
Actual Label = apple vs. Predicted Label = apple
Actual Label = sea vs. Predicted Label = dinosaur
Actual Label = skunk vs. Predicted Label = skunk
Actual Label = streetcar vs. Predicted Label = streetcar
Actual Label = rocket vs. Predicted Label = rocket
Actual Label = lamp vs. Predicted Label = lamp
Actual Label = lion vs. Predicted Label = lion
Actual Label = tulip vs. Predicted Label = tulip
Actual Label = wolf vs. Predicted Label = wolf
Actual Label = rose vs. Predicted Label = rose
Actual Label = orange vs. Predicted Label = orange
Actual Label = rose vs. Predicted Label = rose
```

Discussion for deep neural network :

- The model Accuracy is 64% after preprocessing the training and validation images (Augmentation).
- Tried batch size = 128 instead of batch size = 64 with the same network but it decreased the model accuracy.
- Tried to improve the accuracy using the same network but with batch size = 128 instead of batch size = 64 and with real-time data augmentation but it only increased the training accuracy to about 90% but the model accuracy decreased to 63%.
- Tried to use ResNet50 but it didn't help as the model accuracy didn't change that much.

Links

Shallow neural network code

https://drive.google.com/file/d/1iTN1zvIa8SQFRI23xCNirs80n7tE6CpF/view?usp=sharing

Deep neural network code

https://drive.google.com/file/d/1psILzCroGSVfWn9ZTcn8ut-5-8Gy11fY/view?usp=sharing