ΔΑΔΟΠΟΥΛΟΣ ΜΙΧΑΗΛ

- Databases used: MNIST in intermediate assignment Cifar-10 in the others

Intermediate Assignment(MNIST dataset)

A. Importing first the dataset and preprocess it(reshape and PCA with 2 components)

B. Implementing KNN with sklearn with 3 neighbors accuracy 0.23,fast training

C. Implementing KNN from scratch with 3 neighbors by implementing the following functions:
   - predictKNeighbors that predicts the label of one sample with k nearest neighbors
   - getKNeighbors that returns the labels of k-nearest neighbors of one sample
   - euclideanDistance and sort that were for calculating the euclideian distance between 2 samples and sorting the distances

   Accuracy for the first 300 samples from test set(it was too slow for the whole) was 0.3

D. Implementing NCC with sklearn accuracy 0.33

First Assignment(Cifar-10 dataset)

A. Importing first the dataset and preprocess it:reshape and normalize the data.Depending on the network I used it was needed different reshape
B. Implementing KNN with sklearn with 3 neighbors accuracy 0.06 after PCA(2 components,should be more)
C. Implementing KNN from scratch with 3 neighbors accuracy for the first 300 samples from test set(it was too slow for the whole) was 0.14
D. Implementing NCC with sklearn accuracy 0.2
E. Implementing Neural Network from scratch :

Implementing Neural Network from scratch

Functions used:

- initialize_parameters(layer_dims): for Initialization of parameters using He weight initialization given the nodes of each layer
- Sigmoid(Z),relu(Z),tanh(Z) :activation functions
- linear_forward(A, W, b) :implementation of linear part of a layer's forward propagation
- linear_activation_forward(A_prev, W, b, activation):Implementation of forward propagation for a layer
- L_model_forward(X, parameters,activation):Implementation of forward propagation for L-1 layers with activation "activation" and for the last layer with sigmoid activation
- compute_cost(AL, Y):Implementation of cost function

- relu_local_gradient(dA, cache), sigmoid_local_gradient(dA, cache), tanh_local_gradient(dA,cache):Calculate local gradients for different activation functions given the output of linear layer(cache) and gradient of output of activation of that layer
- linear_backward(dZ, cache):Calculate the gradients for parameters update and for previous layer back propagation
- linear_activation_backward(dA, cache, activation):Implementation of whole backward propagation for a layer
- L_model_backward(AL, Y, caches,activation):Implementation of backward propagation for L-1 layers with activation function and for the last with sigmoid function
- update_parameters(params, grads, learning_rate):Update parameters using gradient descent
- L_layer_model(X, Y, layers_dims,activation, learning_rate = 0.0075, num_iterations = 3000, print_cost=False,):Implementation of a L-layer NN for classification with L-1 layers with "activation" function and the last layer with sigmoid activation that is trained with X and Y for num_iterations iterations with layer_dims numbers of neurons for each hidden with learning_rate

<u>Results:</u> layers [3072, 256, 128, 64, 32, 10] accuracy 0.17 to both train and test set(about 5-6 hours training) and for mnist dataset accuracy 0.43(1 hour and a half)

- **Keras Implementation of same Neural Network build from scratch**

Trained about 10 models with different parameters

Results:Sgd optimiazation and relu gets worse with more iterations.Its worst with sgd optimization than adam, after more iterations

relu is better activation than sigmoid and tanh is the worst.Its perfomance is much better and faster than similar NN from scratch.With adam and more iterations train accuracy improves and slightly improves test accuracy.Train accuracy reached 1.0 and test accuracy about 0.52

- **Keras Implementation of LeNet-5(CNN):**

Results:Test losses are high in every ocassion.Adam optimizer has higher train accuracy and lower train loss but sgd optimizer higher test accuracy and lower test loss ,with more epochs sgd will be better.Is better to have batches size instead of None.Train accuracy 1.0 and test accuracy around 0.63

Second Assignment(Cifar-10 dataset)

- Importing first the dataset and preprocess it:reshape and using MinMaxScaler or StandardScaler
- Visualization of each class
- Implement 3 different SVM models with linear,rbf and poly kernel with sklearn and measure the time of training for each and print correct classifications and false
- Results:

a) StandardScaler is better for SVMs with rbf kernel and MinMaxScaler is better with SVMs with poly and linear kernel

b) RBF kernel:Best value for gamma is 0.1-0.2 and with more and more iterations the model has better accuracy(0.32)

c) Poly kernel:Best value for degree is 2 and with more and more iterations the model has better accuracy(0.33).The same does linear kernel with lower accuracy

d) Best C value is 1.0

Comparing with KNN AND NCC for second and third assignment

- KNN algorithm with 3 neighbors for different n_components at PCA before the training:Best value=32 and accuracy 0.38
- NCC algorithm for different n_components at PCA before the training:best value=256 and accuracy 0.27(for every value accuracy is very close to 0.27)

Third Assignment part 1-Hebbian learning(Cifar-10 dataset)

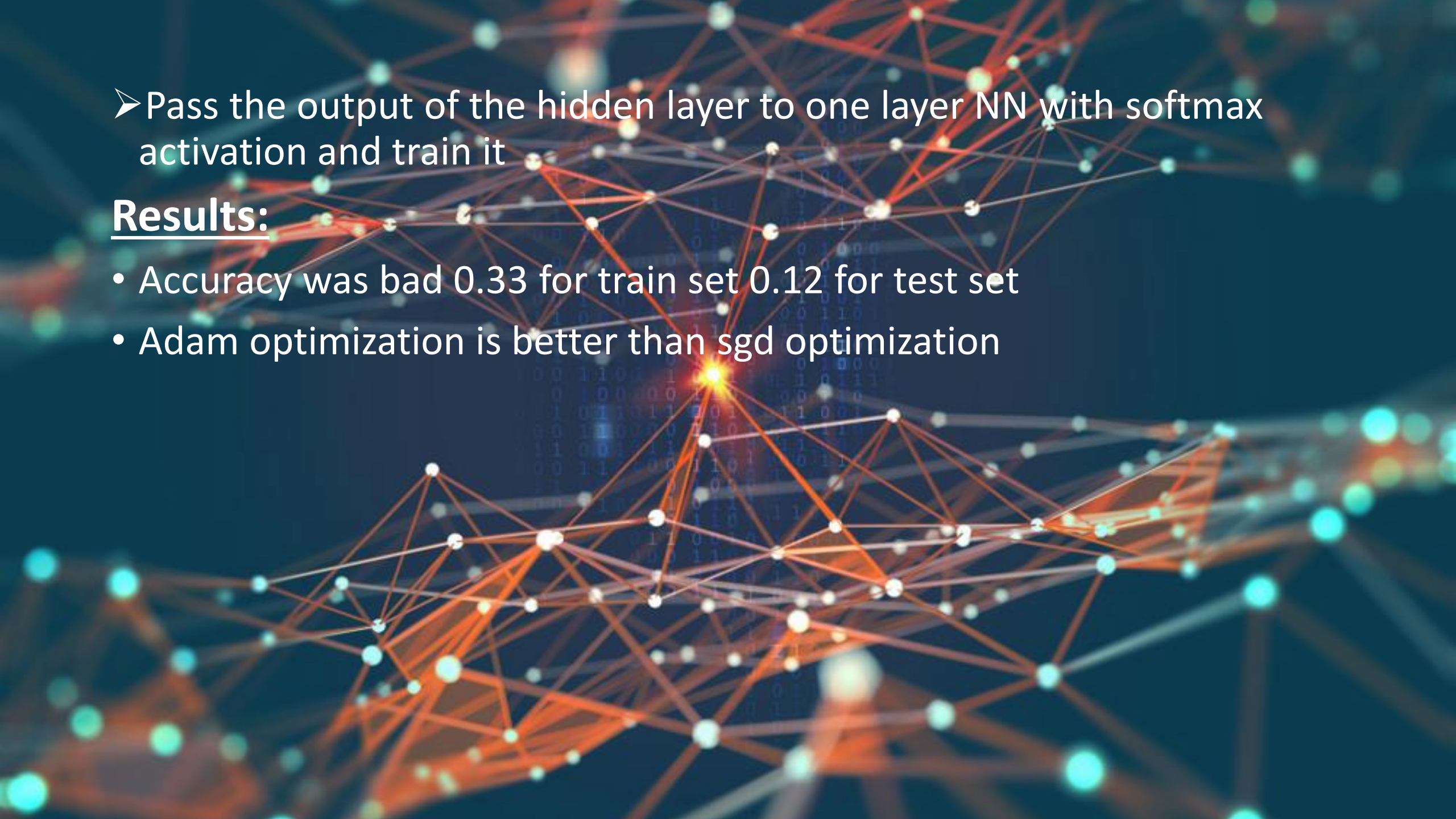
- Importing first the dataset and preprocess it:reshape and Normalization and PCA(n_components=64)
- Visualization of each class
- Implementing Neural Network using Hebbian learning from scratch using the functions(some similar to Neural Network from scratch but a little more simple and certain changes):
  - initialize_parameters(layer_dims):Initialization of parameters to 0
  - Sigmoid(Z),relu(Z),tanh(Z) :activation functions
  - linear_activation_forward(A_prev, W, b, activation):Implementation of forward propagation for a layer

- compute_cost(AL, Y):Compute cross-entropy cost
- L_model_forward_backward_update_parameters(X,Y, params,activation,learning_rate = 0.0075):Implementation of forward propagation for L-1 layers with activation "activation" and for the last layer with sigmoid activation and Update parameters using Hebbian rule L-1 layers and gradient descent for last layer
- L_layer_model(X, Y, layers_dims,activation, learning_rate = 0.0075, num_iterations = 3000,):Implementation of a L-layer NN for classification with L-1 layers with "activation" function and hebbian learning and the last layer with sigmoid activation and gradient descent and printing cost

RESULTS: Didn't really work well,bad accuracy and predicted mostly 0 or 1 only because of high values of weights.I coulnt find much notes and examples of implementations to see what I was lucking and because of that I didn't really tested that much and I implemented part 2

Third Assignment part 2-RBF Neural Network(Cifar-10 dataset)

- Importing first the dataset and reshape it only with out any other preprocess (Normalization didn’t work well)
- Visualization of each class
- Steps of implementation:
  - Find centers for each cluster of the 4096 clusters for the hidden level of rbf with Kmean from sklearn(about 2,5 hours)
  - Find max distance between all centers and stds for Gauss function
  - Calculate ouput for hidden layer with Gauss function for train set(about 1,5 hour).Centers and distances are same for each iteration of training.
  - Calculate ouput for hidden layer with Gauss function for test set(20 minutes)



- Pass the output of the hidden layer to one layer NN with softmax activation and train it

Results:

- Accuracy was bad 0.33 for train set 0.12 for test set
- Adam optimization is better than sgd optimization

NOTES:

- I worked quite a lot but sometimes not that productive,I should have tested some with less samples to see if they work and not spending much time for more samples and having bug.I also maybe had mistakes even I worked hard to find them.
- At the intermediate assignment PCA with n_components=2 was bad idea.Should be higher value
- At first assignment Neural Network from scratch I didn't test it a lot and neither tried many different parameters because it needed much time to implement and train and maybe there was even some mistake to implementation that I didn't spot.
- At second assignment I don't really wrote that much code but I tested many different model and parameters
- At third assignment I didn't count training time
- At third assignment Rbf I didn't tested for different number of center, because for each different number of centers is needed much time for training and at first I spent a lot time figuring out some bugs and the values of weights and maybe even in the end I had mistakes but didn't have much time