CSE574 Introduction to Machine Learning

Programming Assignment 2

Handwritten digits Classification

Report

PROJECT GROUP 16

Rohit Balasayee

Vivek Nagaraju

Sriram Venkataramanan

Problem:

Our task is to implement a Multilayer Perceptron Neural Network and evaluate its performance in classifying handwritten digits. You will use the same network to analyze a more challenging face dataset and compare the performance of the neural network against a deep neural network and a convolutional neural network using the TensorFlow library.

TASKS TO BE IMPLEMENTED

TASK 1

- Implement Neural Network (forward pass and back propagation)
- Incorporate regularization on the weights (_)Use validation set to tune hyper-parameters for Neural
- Network (number of units in the hidden layer and lambda).

TASK 2

Run the deep neural network code and compare the results with normal neural network.

TASK 3

Run the convolutional neural network code and print out the results, for example the confusion matrix.

SOLUTION:

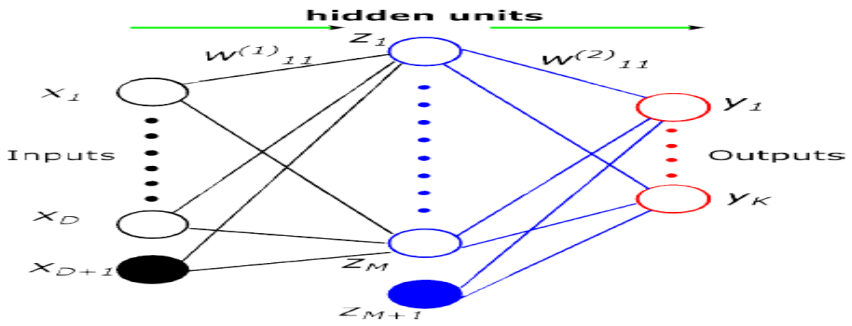
Before we start analysing the data and implementing the code, we need to know some of the basic concepts of Neural Network.

Neural Network:

A neural network is a series of algorithms that endeavours to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. Neural networks can adapt to changing input; so the network generates the best possible result without needing to redesign the output criteria.

Forward pass and Backword pass in Neural Network

- The **forward pass** refers to calculation process, values of the output layers from the inputs data. It's traversing through all neurons from first to last layer.
- A loss function is calculated from the output values.
- And then **backward pass** refers to process of counting changes in weights (de facto *learning*), using gradient descent algorithm (or similar). Computation is made from last layer, backward to the first layer.
- **Backward and forward pass makes together one iteration**.



TASK 1:

We implemented the code in nnscript.py

Data Pre-processing:

Initially we Split the training sets into two sets of 50000 randomly sampled training examples and 10000 validation examples.

Then for all the records we analysed and observed that certain features are repetitive and duplicate i.e. **many features had values which are exactly the same for all data points in the training set. We removed those features which are useless for categorisation.** Our intent was optimising the performance of the neural network by removing those features.

Analysis of the performance of neural network by fine tuning the hyper parameters.

In-order to determine the best accuracy of the neural network over the given MIMST data set, we fine-tuned the following hyper parameters in the code.

- **Setting the regularization hyper-parameter lambda** (lambdaval variable)
- **Setting the number of Hidden units**. [n_hidden = 180 in code]
- **By changing the number of iterations** [opts = {'maxiter': 200} # Preferred value. In code]

We changed these parameters frequently and recorded the accuracy value and time taken at each stage. To determine a favourable regularisation parameter and the number of Hidden units, we kept changing one of the value frequently and other as a constant.

CASE 1:

In this experiment, initially we kept the number of hidden units as 50[given already] as constant and we changed the values of lambda in increments of 5 from 5 to 100. We recorded the values of Time and Accuracy of all three phases and plotted the graphs accordingly.

CASE 2:

In this experiment, we kept the lambda value as a constant [**lambda = 10**] and we changed the number of Hidden units in increments of 30 from 30 to 180. We recorded the values of Time and Accuracy of all three phases and plotted the graphs accordingly.

After analysing all these cases we observed the accuracy of the three phases ie Training Testing and Validation was better at lambda = 10 and Hidden units =180.

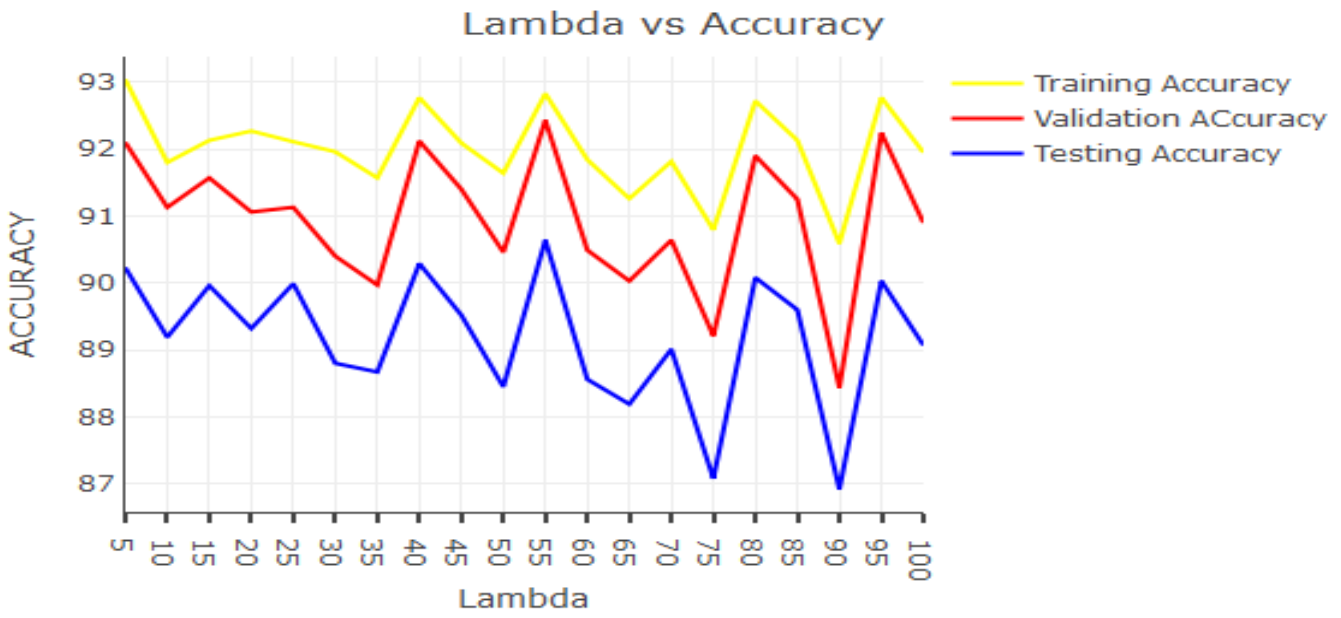
First, we will look at the different types of graphs we plotted.

The 2 cases above mentioned are depicted in 3 Graphs:

1) Lambda Vs Accuracy
2) Number of Hidden Units Vs Accuracy
3) Number of Hidden Units Vs Time taken

GRAPH 1 : Lambda Vs Accuracy

NUMBER OF HIDDEN UNITS =50 [constant]



The values observed in the CSV are:

	A	B	C	D
1	lambda	Training Accuracy	Validation Accuracy	Testing Accuracy
2	5	93.04	92.1	90.23
3	10	91.8	91.13	89.19
4	15	92.13	91.57	89.96
5	20	92.27	91.06	89.32
6	25	92.11	91.13	89.99
7	30	91.96	90.4	88.8
8	35	91.57	89.97	88.67
9	40	92.768	92.12	90.29
10	45	92.09	91.4	89.52
11	50	91.64	90.46	88.45
12	55	92.83	92.43	90.64
13	60	91.842	90.49	88.56
14	65	91.26	90.03	88.19
15	70	91.816	90.64	89.01
16	75	90.795	89.21	87.08
17	80	92.72	91.9	90.08
18	85	92.13	91.25	89.6
19	90	90.59	88.44	86.92
20	95	92.77	92.24	90.03
21	100	91.95	90.9	89.07

EXPLANATION:

As we can see from the above data, we kept the number of hidden units as 50 as a constant and recorded the values for varying values of lambda.

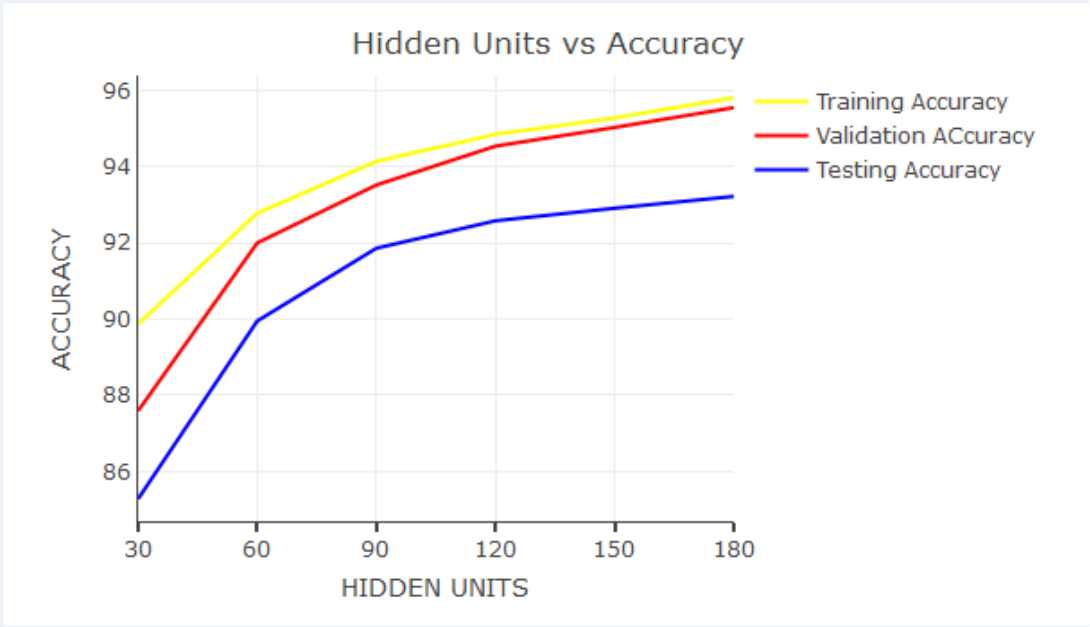
- We can see from the graph that the accuracy increases at certain points of lambda, but it decreases in most of the cases. The Accuracy obtained initially when the lambda is 5 is high in all 3 cases.

- **As we increase the value of lambda, the accuracy is getting decreased bit by bit but not steadily i.e. it doesn't decreases throughout.** We can observe from the graph, as we increase the value of lambda, it's not a guarantee that accuracy increases. Accuracy in all 3 phases increases only at certain stages of lambda.
- **This increases in value of lambda helps avoiding the problem of overfitting.**
- Whereas, we see the accuracy in all three cases i.e. Training, Testing, Validation it decreases as the lambda value increases. Thus it causes overfitting problem.
- **Generally increasing the lambda value helps the network to generalize better**

In-order to select a better lambda value which could address both, we have to choose a smaller value of lambda value 10 preferably.

GRAPH 2: Number of Hidden Units Vs Accuracy

Lambda =10[Constant]



The values observed in the CSV are:

Hidden	Training Accuracy	Validation Accuracy	Testing Accuracy
30	89.87	87.59	85.28
60	92.78	92	89.95
90	94.14	93.52	91.86
120	94.85	94.54	92.58
150	95.28	95.03	92.91
180	95.81	95.55	93.22

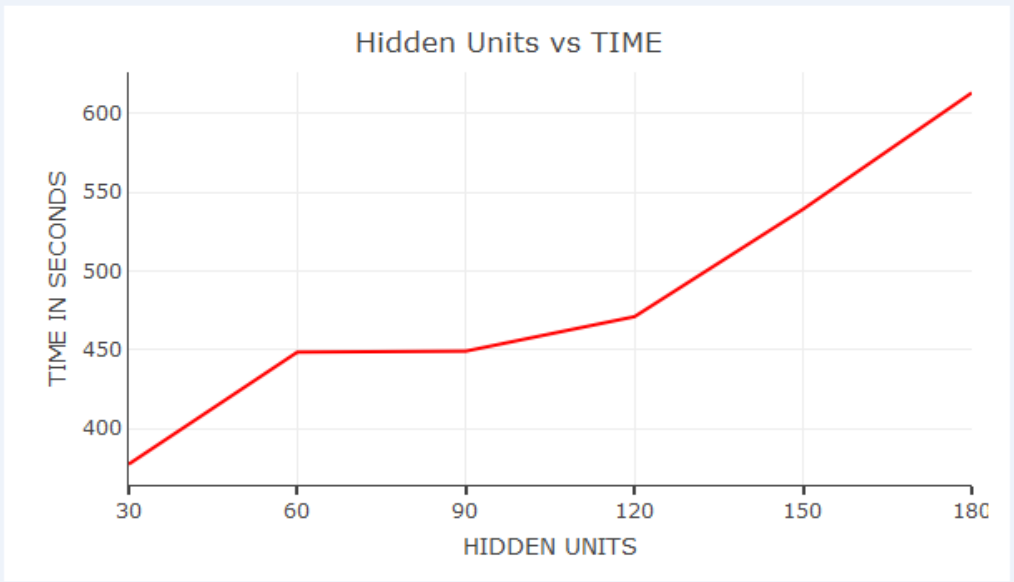
EXPLANATION:

The above graph shows the change in accuracy as we increase the number of Hidden units by keeping the value of Lambda as a constant[10 in our case].

- **Initially the accuracy increased very rapidly**, as we can see in the beginning of the above graph. When we increased the number of Hidden units from 30 to 60, it looks like a **sudden jump in accuracy**.
- **We can see from the graph as the number of Hidden unit's increases, the accuracy of all three phases increases.**
- The highest accuracy in all three phases was achieved at Number of Hidden units =80 i.e. 95.81, 95.55, 93.22 in Training Validation and Testing Phase respectively.

Usually when you increase the number of hidden units, complexity tends to increase and it becomes difficult to fit the data, whereas in this case by keeping lambda as 10 and Hidden units as 180 is helping us to avoid the overfitting problem.

Graph 3: Number of Hidden Units vs TIME



The values observed in the CSV are:

Hidden	Seconds
30	377.2538
60	448.4758
90	449.1051
120	471.0112
150	539.3392
180	613.2772

EXPLANATION:
The above graph shows the change in time as we increase the number of Hidden units by keeping the value of Lambda as a constant[10 in our case].

- Initially we see the time taken was around 380 seconds i.e. 6 minutes approximately when the number of hidden units is 30.
- Then we analyse there is a decent increase in time taken as we move to 60 hidden units which is about 448 seconds i.e. 7.4 minutes.
- Then as we move to 90 and 120 Hidden units, we see the time taken increases slowly.

- The amount of time taken increases briskly as we move to 150 and 180 Hidden units which is about 9 minutes and 10.5 minutes respectively.

Thus as we increase the number of Hidden units, the amount of time taken to complete the process increases.

The increase in time is very obvious, as we need to process the same set of input data over increasing number of Hidden units. This increase in number of Hidden units or features causes more computations on weight and gradient value subsequently.

After full analysis, we observed the neural Network performed better at lambda=10 and Hidden Units=180

TASK 2

Run the deep neural network code and compare the results with normal neural network.
In order to perform task 2 we need to basics about a Deep neural Network.

DEEP NEURAL NETWORK:

A deep neural network is a neural network with a certain level of complexity, a neural network with more than two layers. Deep neural networks use sophisticated mathematical modeling to process data in complex ways.
We implemented the code in **facennScript.py**
We ran the deepnnScript.py for different number of hidden layers and analysed its performance against a simple neural network.

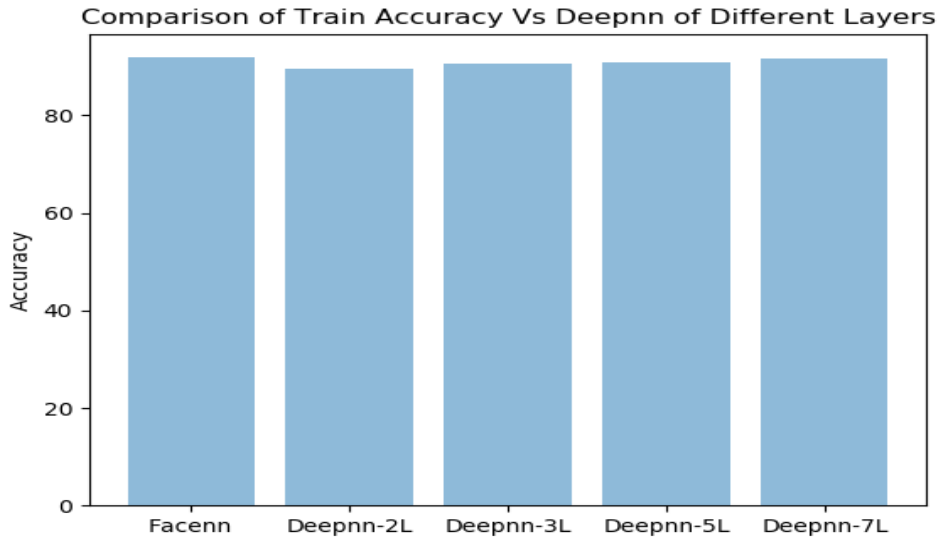
ANALYSIS

We recorded the Training, Validation and Testing accuracy for a simple neural network and Deep neural network and measured the amount of time it took to complete the process.
All these analysis were recorded in a csv file and we plotted it as graphs to get the bigger picture.
The types of graph plotted are:

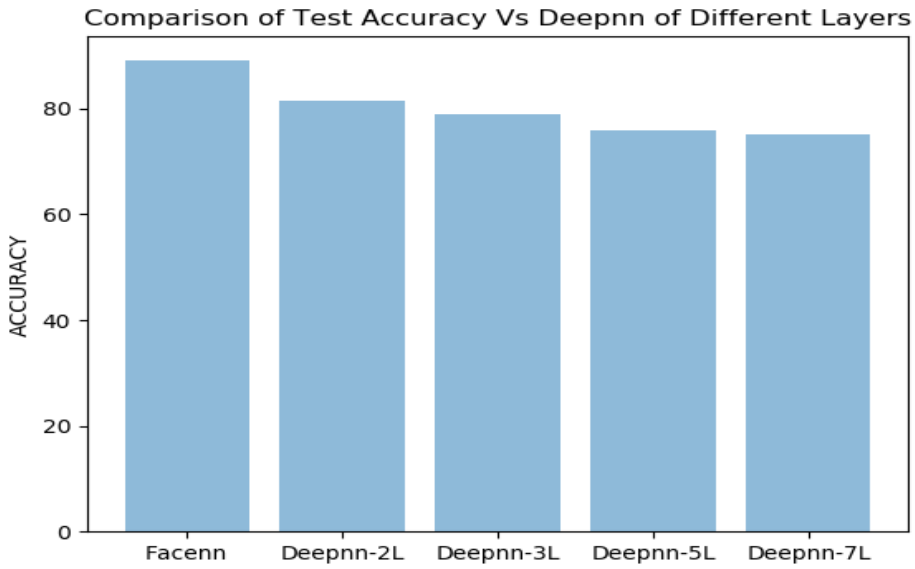
1) Accuracy Measure for Simple Neural network against Deep neural network of 2, 3, 5, 7 Layers.
2) Measurement of Time for Simple Neural network against Deep neural network of 2, 3, 5, 7 Layers.

Graph 1: Accuracy Measure for Simple Neural network against Deep neural network of 2, 3, 5, 7 Layers.

Subgraph 1: Train Accuracy Comparison



Subgraph 2: Test Accuracy Comparison



These are the values recorded in csv file

	TYPE	TRAINING ACCURACY	VALIDATION ACCURACY	TESTING ACCURACY
2	Facenn	91.958	90.9	89.07
3	deepnn layer 2	89.48	80.37	81.302
4	deepnn layer 3	90.74	77.22	78.8
5	deepnn layer 5	90.85	76.36	75.92
6	deepnn layer7	91.75	73.13	75.01

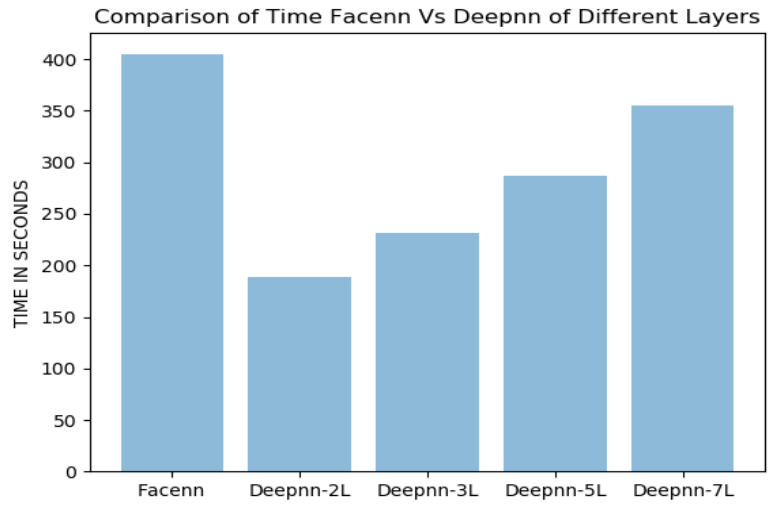
OBSERVATION:

1) **TESTING ACCURACY:**
   - We can see that the testing accuracy decreases as the number of layers gets increased in a deep neural network.
   - The best Testing accuracy was obtained for a simple Neural Network which is 89.07
2) **TRAINING ACCURACY:**
   - We can see that the Training accuracy increases as the number of layers gets increased in a deep neural network.
   - The best Training accuracy was obtained for a simple Neural Network which is 91.75
3) **VALIDATION ACCURACY:**
   We can see from the graph that the validation accuracy decreases as the number of layers increases.

TRAINING ACCURACY VS TESTING ACCURACY

From the graph and from the csv file we can see that in deep neural network, as the number of layers increases, the Training Accuracy increases and the test accuracy decreases. The reason behind this is, as the layer gets increased, the network complexity increases as a result of which it fails to generalise well for unseen data but performs well on Training data.

Graph 2: Measurement of Time for Simple Neural network against Deep neural network of 2, 3, 5, 7 Layers.



The values are recorded in csv file are:

TYPE	TIME IN SECONDS
Facenn	404.7689545
deepnn layer 2	188.9855473
deepnn layer 3	230.9358153
deepnn layer 5	287.2284486
deepnn layer7	354.5011632

OBSERVATION:

- We can see that for a simple Neural Network it took highest time to execute i.e. 404 seconds which is about 6.7 minutes.
- Then as we go to Deep Neural Network we can see that the amount of time it takes to execute increases as we increase the number of hidden layers.
- The facenn runs a neural network with one layer and thus it took more time to compute.
- But in the case of **Deep Neural Network the amount of time increases as we increase the number of layers, this is due to the fact of computing more computations as the layer increases.**

NOTE:
A simple neural network takes more time to compute than a Deep neural network is because of the fact that the Deepnn Code uses Tensor Flow

---

TASK 3:

We ran the code cnnscript.py given to us. In order to run this code, we set up the Tensor Flow environment.
Before analysing the performance of Convolutional neural network, we need to know the basics.

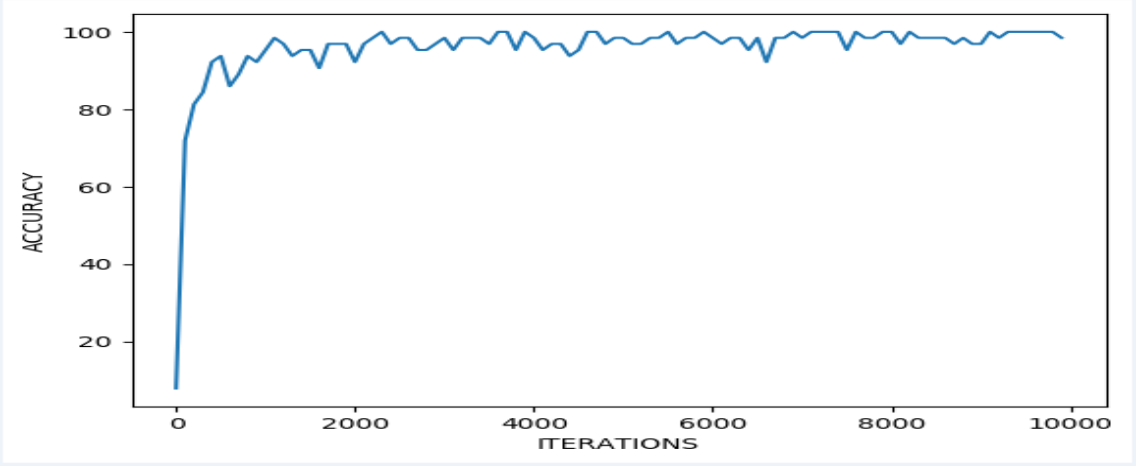
Convolutional neural network

A **Convolutional Neural Network (ConvNet/CNN)** is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

ANALYSIS:

We analysed the performance of CNN over the same data set(face dataset), by plotting the graph of training accuracy and the time it took to run.

GRAPH: TRAINING ACCURACY VS ITERATIONS OF A CNN



THE OUTPUT OBTAINED BY RUNNING THE CODE GIVEN:

```
Accuracy on Test-Set: 8.9% (892 / 10000)
Optimization Iteration:      1, Training Accuracy:   7.8%
Time usage: 0:00:01
Accuracy on Test-Set: 8.9% (892 / 10000)
Time usage: 0:00:08
Accuracy on Test-Set: 66.2% (6617 / 10000)
Optimization Iteration:    101, Training Accuracy:  71.9%
Optimization Iteration:    201, Training Accuracy:  81.2%
Optimization Iteration:    301, Training Accuracy:  84.4%
Optimization Iteration:    401, Training Accuracy:  92.2%
Optimization Iteration:    501, Training Accuracy:  93.8%
Optimization Iteration:    601, Training Accuracy:  85.9%
Optimization Iteration:    701, Training Accuracy:  89.1%
Optimization Iteration:    801, Training Accuracy:  93.8%
Optimization Iteration:    901, Training Accuracy:  92.2%
Time usage: 0:01:08
Accuracy on Test-Set: 93.2% (9321 / 10000)
Optimization Iteration:   1001, Training Accuracy:  95.3%
Optimization Iteration:   1101, Training Accuracy:  98.4%
```

```
Optimization Iteration:   8701, Training Accuracy:  96.9%
Optimization Iteration:   8801, Training Accuracy:  98.4%
Optimization Iteration:   8901, Training Accuracy:  96.9%
Optimization Iteration:   9001, Training Accuracy:  96.9%
Optimization Iteration:   9101, Training Accuracy: 100.0%
Optimization Iteration:   9201, Training Accuracy:  98.4%
Optimization Iteration:   9301, Training Accuracy: 100.0%
Optimization Iteration:   9401, Training Accuracy: 100.0%
Optimization Iteration:   9501, Training Accuracy: 100.0%
Optimization Iteration:   9601, Training Accuracy: 100.0%
Optimization Iteration:   9701, Training Accuracy: 100.0%
Optimization Iteration:   9801, Training Accuracy: 100.0%
Optimization Iteration:   9901, Training Accuracy:  98.4%
Time usage: 0:12:00
Accuracy on Test-Set: 98.8% (9878 / 10000)
```

OBSERVATION:

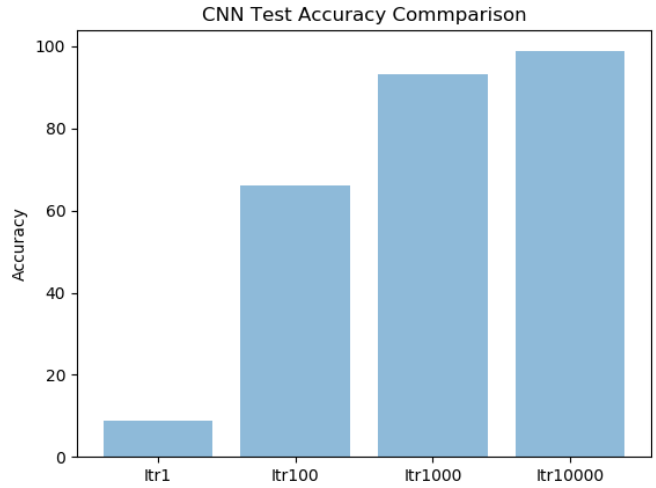
From the graph and from the output we can observe the following:

TRAINING ACCURACY

- Training accuracy initially after a single iteration was around 7.8%
- Training accuracy after 100 iterations was about 71.9%
- Training accuracy becomes stays less than 100 till about 400 iterations.

- After 400 iterations, we can observe from the graph the training accuracy becomes constant and reaches 100% at certain stages.

TEST ACCURACY



As you can see from the graph increase Test accuracy increases as the Iterations increases.
Final Test Accuracy obtained after 10000 iterations : 98.8
TIME TAKEN TO RUN THE CODE: It took almost 12 mins to complete the full process.

REASONING:

The accuracy is very high compared to other codes such as Simple Neural network or Deep Neural network is because of the fact CNN is best in visual recognition. It learns part by part and eventually builds the overall network.

CNN also takes less computation time than other MLP, because of a concept called Parameter Sharing

---

CONCLUSION

- **As the number of Hidden units increases, the computational time increases and the accuracy also increases. A trade off should be made between the two and in our case, highest accuracy was obtained at Lambda=10 and Hidden units =180**
- **As the number of layer gets increased in Deepnn, the network complexity increases as a result of which it fails to generalise well for unseen data but performs well on Training data.**
- **The accuracy is very high in CNN is because of the fact CNN is best in visual recognition.**
