Deep Learning (CS F425)

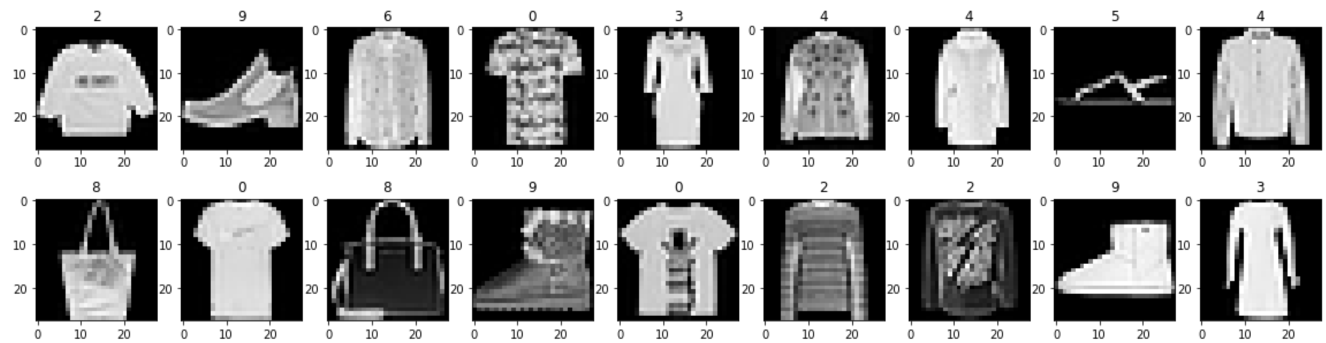
Assignment – 1

Comparative study of Artificial Neural Network models

Achyut Bajpai 2019A8PS0499H
Nandan H R 2019A7PS0164H
Durba Satpathi 2019A7PS0972H

1. Introduction

The given problem is an image classification problem, with 10 labels.




```
Training Data size (60000, 785)
Testing Data size (10000, 785)
```

The labels were converted to one-hot encodings.
The input layer contains 784 (28*28) nodes.
Since we are doing one of k classification (k=10), i.e the output layer contains k nodes and the target variable is a k dimensional vector of which exactly one component is 1 and remaining are zero. Consequently we use softmax activation function in the output layer.

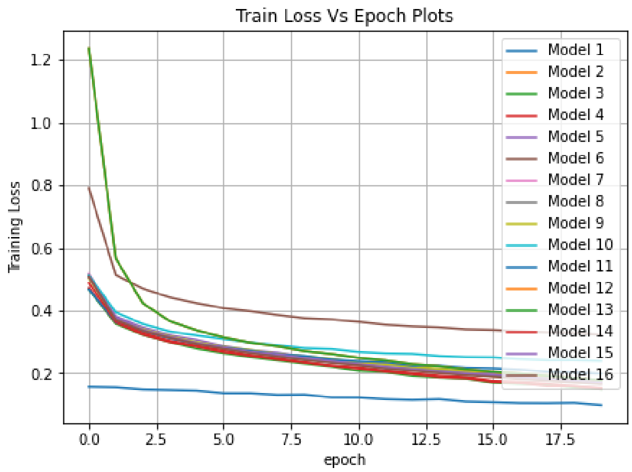
2. Results of the various models

Model No	Hidden Layers	Activation Function	Loss Function	Number of Parameters	Testing Accuracy	Training Loss	Testing Loss
1	0 (baseline)	none	categorical_crossentropy	7850	0.8390	0.2027	0.5223
2	1,[256]	relu	categorical_crossentropy	203,530	0.8909	0.1666	0.3270
3	2,[512,256]	relu	categorical_crossentropy	535,818	0.9021	0.1492	0.3354

4	3,[512,512,256]	relu	categorical_crossentropy	798,474	0.8994	0.1614	0.3519
5	5,[512,512,512,512,512]	relu	categorical_crossentropy	1,457,674	0.9028	0.1378	0.3235
6	5,[16,16,16,16,16]	relu	categorical_crossentropy	13,818	0.8655	0.3223	0.3787
7	5,[512,256,128,64,32]	relu	categorical_crossentropy	576,810	0.8926	0.1693	0.3511
8	5,[512,256,128,64,32]	relu	kl_divergence	576,810	0.8945	0.1712	0.3500
9	5,[256,256,256,256,256]	relu	categorical_crossentropy	466,698	0.9014	0.1476	0.3249
10	5,[512,256,128,64,32]	tanh	categorical_crossentropy	576,810	0.8920	0.1785	0.3515
11	2,[512,256]	tanh	categorical_crossentropy	535,818	0.8912	0.1567	0.3126
12	5,[512,256,128,64,32]	sigmoid	categorical_crossentropy	576,810	0.8919	0.1810	0.3551
13	2,[512,256]	sigmoid	categorical_crossentropy	535,818	0.8885	0.1573	0.3143
14	3,[256,256,256]	relu	categorical_crossentropy	335,114	0.8975	0.1591	0.3258
15	5,[512,256,128,64,128]	relu	categorical_crossentropy	584,010	0.9089	0.1609	0.3138
16	4,[500,200,75,100]	relu	categorical_crossentropy	516,385	0.8987	0.1633	0.3142

2. Plots

2.1 Training Loss



2.2 Testing Loss

2.3 Testing Accuracy

3. Comparative Study

3.1. Number of parameters / Number of Nodes in the hidden layers

Model No	Hidden Layers	Activation Function	Loss Function	Number of Parameters	Testing Accuracy	Training Loss	Testing Loss
6	5,[16,16,16,16,16]	relu	categorical_crossentropy	13,818	0.8655	0.3223	0.3787
7	5,[512,256,128,64,32]	relu	categorical_crossentropy	576,810	0.8926	0.1693	0.3511
5	5,[512,512,512,512,512]	relu	categorical_crossentropy	1,457,674	0.9028	0.1378	0.3235

- Keeping the number of hidden layers constant, the total number of nodes is proportional to the total number of parameters.
- As the number of parameters increases both train and test losses of the models were observed to decrease, but the decrease in train loss is more as compared to the test loss
- This might be because with more parameters, we can represent more complex functions, resulting in the model fitting the training data better.
- However, with a huge number of parameters, the model may not generalize well i.e may not fit the testing data as good as it fits the training data, and might result in overfitting
- Also with increase in number of the parameters, training time of the model increases

3.2. Number of Hidden Layers

Model No	Hidden Layers	Activation Function	Loss Function	Number of Parameters	Testing Accuracy	Training Loss	Testing Loss
2	1,[256]	relu	categorical_crossentropy	203,530	0.8909	0.1666	0.3270
14	3,[256,256,256]	relu	categorical_crossentropy	335,114	0.8975	0.1591	0.3258
9	5,[256,256,256,256,256]	relu	categorical_crossentropy	466,698	0.9014	0.1476	0.3249

- As the number of hidden layer increases performance of the models improved,

- This might be because we are able to extract more abstract features and learn better representations of features using more hidden layers.
- DL Algorithms attempt to learn multiple levels of representations by using a hierarchy of multiple layers.
- Higher layers of representations amplify aspects of input important for dicrimination and suppress irrelevant variations.
- However too many hidden layers result in huge number of parameters, and training time of the model increases

3.3. Activation Functions

Model No	Hidden Layers	Activation Function	Loss Function	Number of Parameters	Testing Accuracy	Training Loss	Testing Loss
7	5,[512,256,128,64, 32]	relu	categorical_crossentropy	576,810	0.8926	0.1693	0.3511
10	5,[512,256,128,64, 32]	tanh	categorical_crossentropy	576,810	0.8920	0.1785	0.3515
12	5,[512,256,128,64, 32]	sigmoid	categorical_crossentropy	576,810	0.8919	0.1810	0.3551
3	2,[512,256]	relu	categorical_crossentropy	535,818	0.9021	0.1492	0.3354
11	2,[512,256]	tanh	categorical_crossentropy	535,818	0.8912	0.1567	0.3126
13	2,[512,256]	sigmoid	categorical_crossentropy	535,818	0.8885	0.1573	0.3143

- ReLu activation function was seen to perform slightly better than tanh and sigmoid activations.
- The gradients of tanh and sigmoid saturate to zero at higher values and thus the weights stop updating significantly. However ReLu has a constant gradient for all positive values and does not face this problem
- ReLu function is faster to compute as compared to sigmoid, tanh and sigmoid functions and helps in reducing the training time by a small amount. Also when we use ReLu function for activation, only the nodes receiving positive inputs get activated, thus making it more computationally efficient as compared tanh and sigmoid activations.

- Tanh is generally preferred over sigmoid function because its gradients are not restricted to a specific direction and that it is zero centred

3.3. Loss Functions

$$
\begin{aligned}
D_{\mathrm{KL}}(p|q) &= \textstyle\sum_i p_i \log \frac{p_i}{q_i} \\
&= \textstyle\sum_i (-p_i \log q_i + p_i \log p_i) \\
&= -\textstyle\sum_i p_i \log q_i + \sum_i p_i \log p_i \\
&= -\textstyle\sum_i p_i \log q_i - \sum_i p_i \log \frac{1}{p_i} \\
&= -\textstyle\sum_i p_i \log q_i - H(p) \\
&= \textstyle\sum_i p_i \log \frac{1}{q_i} - H(p)
\end{aligned}
$$

where $D_{KL}(p|q)$ represents the KL divergence loss of probability distribution p with respect to q.

$$H(p, q) = \sum_i p_i \log \frac{1}{q_i}.$$

where $H(p, q)$ represents the cross entropy loss between probability distributions p and q.

In one of k classification where the output is a single class, H(p) becomes zero and the KL divergence loss becomes equal to the cross entropy loss. And hence we get similar results while using these loss functions. The slight difference in the results is due to randomization.

Model No	Hidden Layers	Activation Function	Loss Function	Number of Parameters	Testing Accuracy	Training Loss	Testing Loss
7	5,[512,256,128,64, 32]	relu	Categorical_ rossentropy	576,810	0.8926	0.1693	0.3511
8	5,[512,256,128,64, 32]	relu	KL divergence	576,810	0.8945	0.1712	0.3500

4. Best Performing Model

The autoencoder type architecture gave the best results among all the other models.

Model No	Hidden Layers	Activation Function	Loss Function	Number of Parameters	Testing Accuracy	Training Loss	Testing Loss
15	5,[512,256,128,64,128]	relu	categorical_crossentropy	584,010	0.9089	0.1609	0.3138
16	4,[500,200,75,100]	relu	categorical_crossentropy	516,385	0.8987	0.1633	0.3142

In autoencoder type models the input is first compressed into a latent space representation. Here the most abstract features are obtained and the important aspects of inputs are amplified for and irrelevant features are suppressed. The outputs are then reconstructed from this latent space representation Thus the autoencoder type architecture tends to give better results as compared to other models