

Selected Topics in Computer Science

Deep Learning

Assignment - 1

Project Report

Team 5
Team Members

Raunak Mantri - 2017B5A71340H
Adesh Pradhan - 2017B3A70960H
Anushray Mathur - 2017A7PS1570H

Deep Learning Models

Default Parameters for all Models-

1. Batch size = 128
2. Normalization = [0,1]
3. Optimizer = SGD
4. Epochs = 50
5. Metrics = 'accuracy'

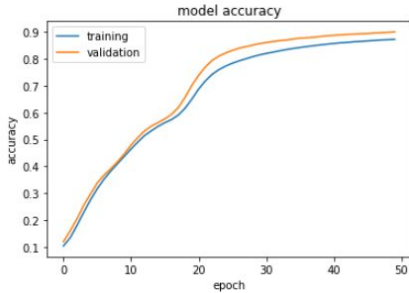
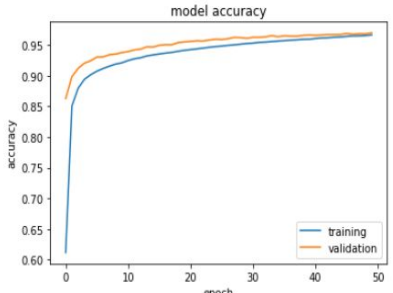
We tried to look at different optimisers like SGD and Adam. In most of the models Adam gave a better performance. But we looked at SGD as it was familiar with most of us and we could interpret the results.

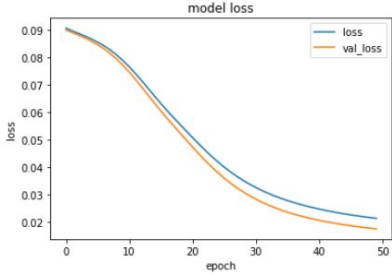
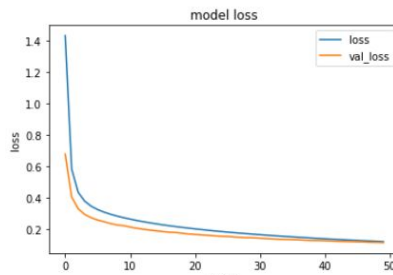
(A) Different Loss Functions

- Architecture:

Activation - ReLu
Layers - [64, 64]
Output Activation - SoftMax

- Observation:

Metrics/Loss Function	MSE (Model No. 1)	Categorical Cross Entropy (Model No. 2)
Accuracy	 <p>Accuracy = 88.08%</p>	 <p>Accuracy = 96.05%</p>

Loss	 <p>Loss = 0.0199</p>	 <p>Loss = 0.128</p>
------	--	---

- Explanation:

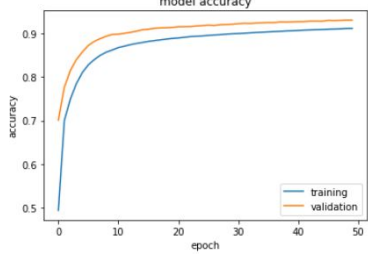
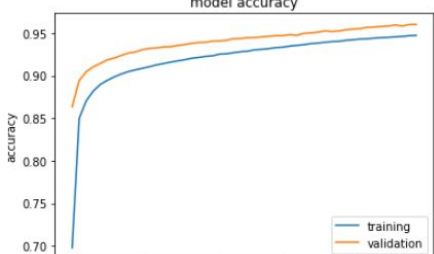
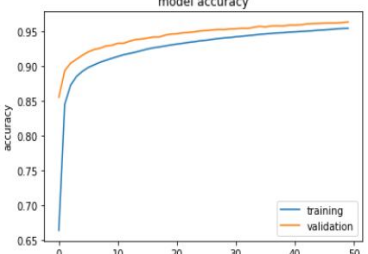
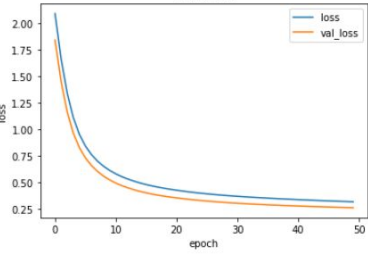
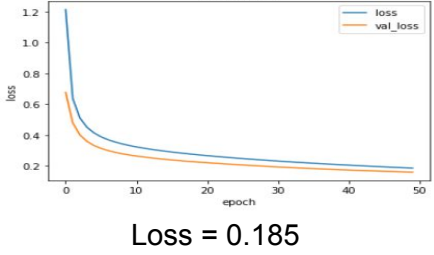
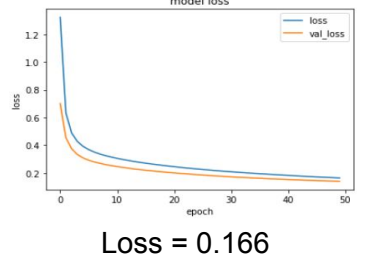
1. When comparing the use of different cost functions for the same parameters, we find that Categorical Cross Entropy performs significantly better than MSE.
2. Since this is an example of a classification problem, Categorical Cross Entropy was expected to perform better. This is due to the fact that Categorical Cross Entropy gives larger gradients when the prediction is wrong and small gradients when prediction is almost correct. Whereas the error provided by MSE is much more suited to regression problems.

(B) Different Activation Functions

- Architecture:

Loss Function - Categorical Cross Entropy
Layers - [64]
Output Activation - SoftMax

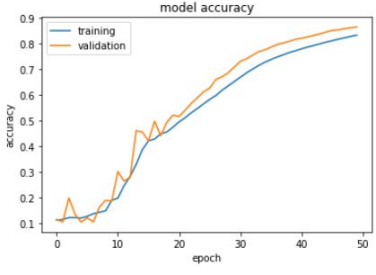
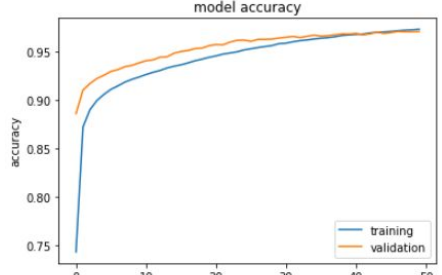
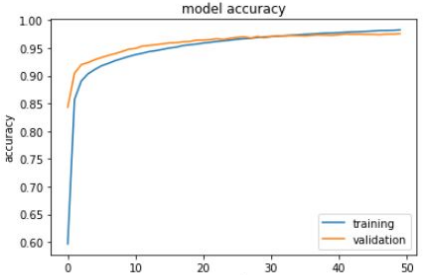
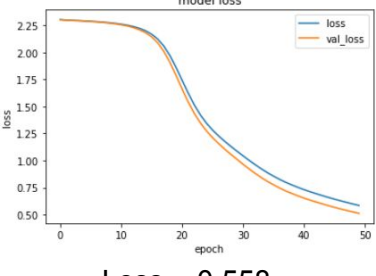
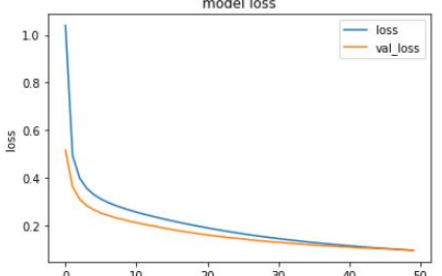
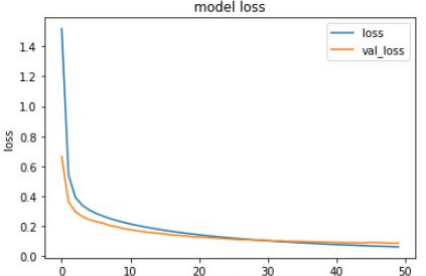
- Observation:

	Sigmoid (Model No. 3)	Tanh (Model No. 4)	ReLu (Model No. 5)
Accuracy	 <p>Accuracy = 91.83%</p>	 <p>Accuracy = 94.66%</p>	 <p>Accuracy = 95.21%</p>
Loss	 <p>Loss = 0.297</p>	 <p>Loss = 0.185</p>	 <p>Loss = 0.166</p>

- Architecture:

Loss Function - Categorical Cross Entropy
 Layers - [128,128,128]
 Output Activation - SoftMax

- Observation:

	Sigmoid (Model No. 6)	Tanh (Model No. 7)	ReLu (Model No. 8)
Accuracy	 <p>Accuracy = 84.18%</p>	 <p>Accuracy = 96.74%</p>	 <p>Accuracy = 97.09%</p>
Loss	 <p>Loss = 0.558</p>	 <p>Loss = 0.112</p>	 <p>Loss = 0.0952</p>

- Explanation:

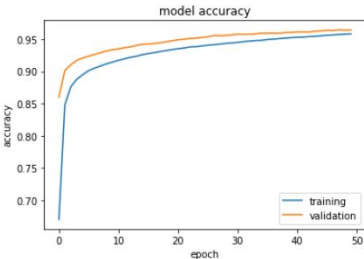
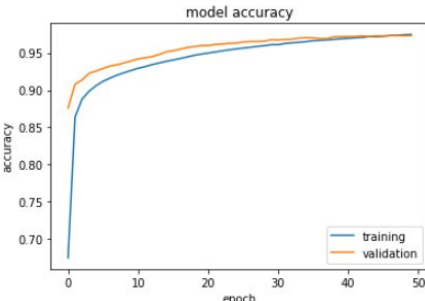
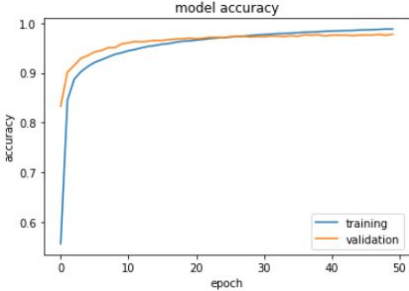
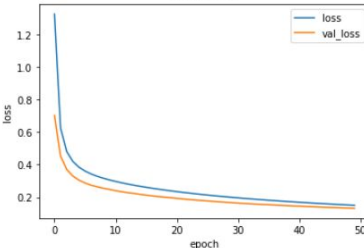
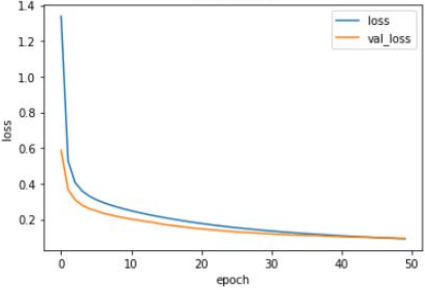
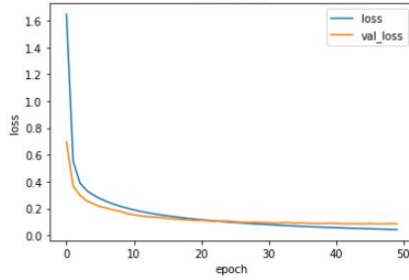
1. The performance seen is ReLu > Tanh > Sigmoid
2. This can be justified from the fact that ReLu does not undergo saturation while tanh and sigmoid saturate.
3. Also, tanh has a better performance than sigmoid since it is zero-centred. Non zero centeredness is undesirable since neurons in later layers of processing would be receiving data that is not zero-centered. This has implications on the dynamics during gradient descent, because if the data coming into a neuron is always positive (e.g. $x > 0$ elementwise in $f = wTx + b$), then the gradient on the weights w will during backpropagation become either all be positive, or all negative (depending on the gradient of the whole expression f). This could introduce undesirable zig-zagging dynamics in the gradient updates for the weights.
4. Sigmoid units tend to saturate which is why it is never used in hidden layers; as it provides outputs between 0 and 1, it is suited as the output layer for binary classification problems only.

(C) Different Hidden layers

- Architecture:

Loss Function - Categorical Cross Entropy
Activation Function - ReLu
Output Activation - SoftMax

- Observation:

	128 (Model No. 9)	128,128 (Model No. 10)	128,128,128,128 (Model No. 11)
Accuracy	 <p>Accuracy = 95.7%</p>	 <p>Accuracy = 96.77</p>	 <p>Accuracy = 97.35</p>
Loss	 <p>Loss = 0.153</p>	 <p>Loss = 0.109</p>	 <p>Loss = 0.0918</p>

- Explanation:

1. It can be observed clearly that as the number of layers increase in the above examples, the accuracy increases.
2. However, having too many layers can lead to some difficulties such as overfitting. This happens as the model tries to follow each and every irregularity that might be present in the training set and thus lose generality.

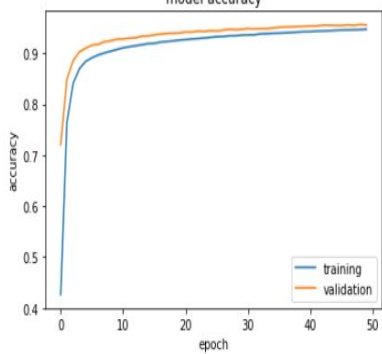
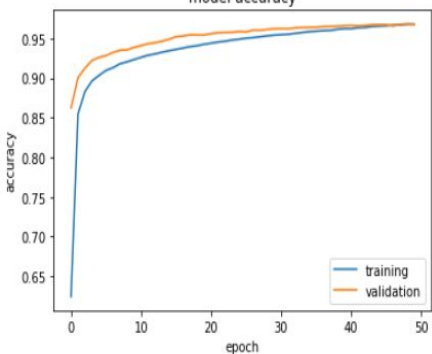
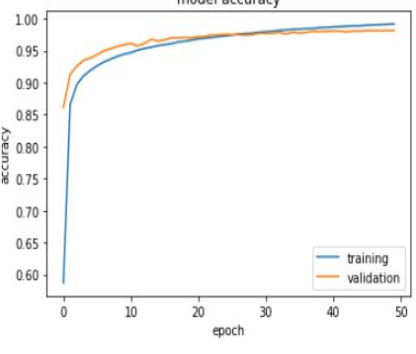
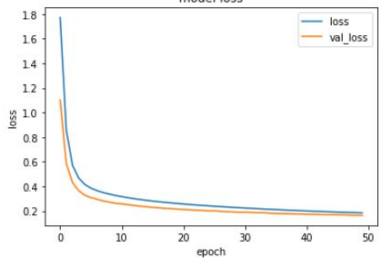
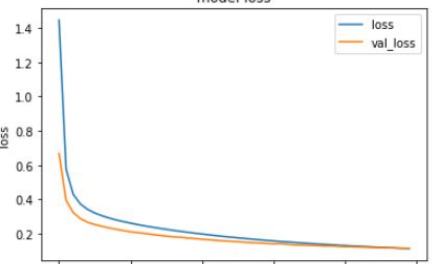
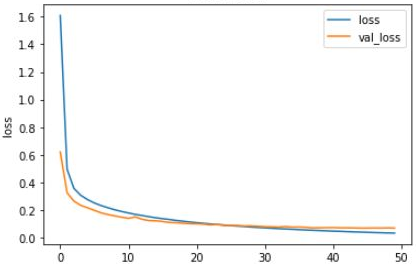
- We can see that towards the end of Model 11, the validation accuracy becomes stagnant/starts dropping but the training accuracy still increases. Thus we can say that at this point the model has just started overfitting to the training set.

(D) Increasing Neurons

- Architecture:

Loss Function - Categorical Cross Entropy
 Activation Function - ReLu
 Output Activation - SoftMax

Observation:

	16, 16 (Model No. 12)	64, 64 (Model No. 13)	256, 256 (Model No. 14)
Accuracy	 <p>Accuracy = 94.52%</p>	 <p>Accuracy = 96.4%</p>	 <p>Accuracy = 97.62%</p>
Loss	 <p>Loss = 0.191</p>	 <p>Loss = 0.123</p>	 <p>Loss = 0.0755</p>

- Explanation:

1. It can be observed that when all other parameters are constant, increasing the number of neurons in the layers will result in better accuracy.
2. But as the case is with the number of layers, having too many neurons will result in having many parameters for the model and can lead to overfitting.

Machine Learning Algorithms

(A) Decision Tree

- Architecture:

```
Max_depth = 10  
min_samples_leaf = 1  
Random_state = 0 (To provide consistency in results)
```

- Observation:

	Test	Training
Accuracy	84.22	87.49

- Rationale:

The first parameter indicates how deep the tree can be. The deeper the tree, more splits it has and thus it can capture more information about the data.

Max_Depth	Test Accuracy
2	9.87
10	84.22
20	87.73
32	87.87

However, increasing depth may lead to overfitting (High training accuracy and low test accuracy).

(B) SVM

- Architecture:

Kernel = Radial Basis Function Regularization parameter = 10 Gamma(Kernel coefficient) = 0.01

- Observation:

	Test
Accuracy	98.21

- Description:

Suppose there are n number of data points which are features of our dataset. These points are plotted in an n -dimensional graph or space. After classification, these features are divided by a hyperparameter line which separates the two plains and helps us find the best one.

There are many possible criteria for finding the optimum hyperplane. Our objective is to find a plane in which has the maximum distance between the data points of both the classes. The dimension of a hyperplane is directly proportional to the number of features. If we input only 2 features, then the hyperplane is just a line. If it's 3 or more, then the hyperplane becomes a two-dimensional plane. It becomes more complex when it exceeds 3.

These data points are support vectors that are closer to the hyperplane and influence the position and orientation of the hyperplane. With the help of these vectors, we maximize the margin of the classifier. If we change or delete support vectors, it will change the position of the hyperplane. These points help us build our SVM model.

(C) Multi-Class Logistic Regression

- Architecture:

Batch Size = 100 No. of Iteration = 3000 Epochs = 6 Learning rate = 0.001
--

- Observation:

	Test
Accuracy	83.09
Loss	0.96

- Description:

Logistic regression is one of the most fundamental and widely used Machine Learning Algorithms. Logistic regression is not a regression algorithm but a probabilistic classification model.

Classification in Machine Learning is a technique of learning, where an instance is mapped to one of many labels. The machine learns patterns from data in such a way that the learned representation successfully maps the original dimension to the suggested label/class without any intervention from a human expert.

Logistic regression has a sigmoidal curve.

The idea in logistic regression is to cast the problem in the form of a generalized linear regression model.

$$\hat{y} = \beta_0 + \beta_1 x_1 + \dots + \beta_n x_n$$

where \hat{y} = predicted value, x = independent variables and the β are coefficients to be learned.

Thus, the logistic link function can be used to cast logistic regression into the Generalized Linear Model.

(D) Random Forest

- Architecture:

```
N_jobs = -1
N_estimator = 10
min_samples_leaf = 1
```

- Observation:

	Test	Training
Accuracy	86.34	99.17

- **Description**
Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. In Decision Trees, for predicting a class label for a record we start from the root of the tree. We compare the values of the root attribute with the record's attribute. On the basis of comparison, we follow the branch corresponding to that value and jump to the next node.

Here we can see a clear case of overfitting as the training accuracy is quite higher than test accuracy.

Conclusion

From the above examples we can clearly see that a well formed deep learning model performs significantly better than most of the traditional ML classification models. Deep Learning Models perform better than classical machine learning models.

The reasons may be-

1. Scales effectively with data
2. No need for feature engineering: Classical ML algorithms often require complex feature engineering. A dimensionality reduction might then be done for easier processing. Finally, the best features must be carefully selected to pass over to the ML algorithm. There's no need for this when using a deep network as one can just pass the data directly to the network and usually achieve good performance right off the bat.
3. Deep neural networks can create a lot of decision boundaries than classical models even with less data.