Fall 2018

Matthew Dobbin

Northwestern University

MSDS 458 Artificial Intelligence

Assignment 1 – Classifier Neural Networks

Contents

1.0 Introduction

This report describes the process of designing, training and assessing a multilayer perceptron neural network models that were trained using the backpropagation method. The purpose of these models was to understand the impact of the hyperparameters and to understand how the nodes in a simple single hidden layer network have learned to represent features within the input data. The models learn to classify six capital letters; A, D, O, E, F and Q.

The final part of the report explores a real world application of neural networks; using neural network models for hand written digit recognition. All tables and images referenced in the report are contained within the Appendix. The author is still relatively new to Python due to taking the 'R' based PRED410, 411, 422 courses.

2.0 Multilayer Perceptron Neural Network

The neural network that was implemented uses a three layer network consisting of an input layer, a single hidden layer and an output layer. The capital letters were represented using a 9x9 input grid. Printouts of the six capital letters from the training data are shown in Figure 1. The 9x9 input grid was flattened into an 81x1 array so that each "pixel" was an input node into the model. The sigmoid activation function was used as the transfer function for both the hidden and output layers.

2.2 Learning Rate

The model was trained using backpropogation, which is an algorithm that adjusts the weight of the neurons based on the gradient of the loss function. The sum of the squared errors (SSE) between the predicted values and the desired values was used as the loss function. The learning rate of the model was adjusted by changing the value of eta. Eta scales the amount of change to the connection weights.

Figure 2 shows a plot of the loss function versus the number of training iterations. In each plot the eta value has been changed to show how the parameter affects model training. The training of the model stops once the SSE value is below a threshold value. The threshold for this case was selected as 0.005. From the plots it can be seen that it requires more iterations to train the model at lower eta values.

2.3 Hidden Node Analysis

The letters A, D, O, E, F and Q were selected as [E, F] and [D, O, Q] share similar shape characteristics while the shape of A is quite different. It was thought that there may be a discernible pattern between the values of the hidden layer nodes for the respective letters. For example, it was hypothesised that the horizontal middle line for E and F may be represented by a particular hidden node.

Two different models were evaluated. The first model used six nodes for the single hidden layer and the second model used 12 nodes in the single hidden layer. The hidden layer nodes and output values for the model with six hidden layers nodes are shown in Table 1. The heat map shown in Figure 3 was created to visualise the hidden layer node activation values for each letter.

For the letter A, there are high activation values for hidden nodes 0 and 4 and it is the only letter that has a high activation level on node 4. Letters E and F have similar activation values for node 2 however the values for the other hidden nodes are not similar.

The heat map for the model with twelve hidden layers nodes is shown in Figure 4. Interestingly, for the letter E, 9 of the 12 hidden nodes have high activation values. Node 11 has a high activation level for all of the letters and node 7 does not have an activation value greater than 0.3 for all the letters.

For these two models it appears as though that the features that our visual systems detect such as horizontal and circular lines were not necessarily represented as the features the neural network sees. The neural network is seeing on/off activations in an 81-element long vector.

3.0 Real World Application

There are numerous practical applications for letter and digit pattern recognition. One practical example for digit recognition has been to use a robot to sort mail based on reading the typed or handwritten addresses on envelopes. A system offered by SOLYSTIC that uses neural networks can achieve over 98% accuracy (Solystic, 2018).

An advantage of a robotic solution instead of a human is that the robot can sort the mail at a faster rate, can operate 24 hours a day and does not require sick or annual leave. A data driven approach would be used to train the algorithm that the robot uses to sort the mail. The following report sections detail how a neural network can be used for digit pattern recognition.

3.1 MNIST Data Set

The MNIST data set was assembled by the National Institute of Standards and Technology in the 1980s. It contains 28x28 pixel grayscale images of handwritten numbers between zero and nine. The training and test sets contain 60,000 and 10,000 images respectively. The first four images from the training data are shown in Figure 5 and the distribution of the numbers in the dataset are shown in Table 2. The data set requires minimal pre-processing prior to using in a neural network model. The 28x28 input array is reshaped so that the input into the neural network has 784 input nodes and the values are scaled so that they are between zero and one.

3.2 Classification Modelling

Four different models were trialled to compare the classification accuracy and computational time of models with different network architectures. All of the models used the relu activation function to transform the input data. The softmax activation function was used for the final layer as it returns the probability scores for the ten output nodes (numbers 0-9). The loss function for the models was chosen as "categorical_crossentropy". This loss function measures the "distance between the probability distribution output by the network and the true distribution of the labels. By minimizing the distance between these two distributions, you train the network to output some thing as close as possible to the true labels" (Chollet, 2017).

The first three models have a single hidden layer. The number of nodes in the hidden layer was varied, using 100, 250 and 500 nodes. The fourth model has two hidden layers, each with 250 nodes. The models were trained for 20 epochs and used a batch size of 512.

3.3 Model Performance

The performances of the models are shown in Table 3 as well in Figure 6, Figure 7, Figure 8 and Figure 9 which graph the training and validation performance. On all of the models, the prediction accuracy on the validation (test) set starts to level out between 10-15 epochs. After this the models are over-fitting the training data. The model with a single hidden layer and the least number of hidden layer nodes had the lowest classification accuracy with a test set score of 97.64%. However it took the shortest time to train the model with a computational time of 22 seconds.

The best performing model was the single hidden layer with 500 nodes. It achieved a test set accuracy of 98.26%. For this case, a model with a single hidden layer with 500 nodes out performs a model with two hidden layers each with 250 nodes.

4.0 Conclusions

The multilayer perceptron neural network that classifies capital letters found that the features that our visual systems detect, such as horizontal lines were not necessarily represented as the features the neural network sees. The neural network is seeing on/off activations in an 81-element long vector.

There are a variety of applications where using neural networks for letter or digit recognition can be used to reduce costs (human overheads) and increase productivity for businesses. Manager's need to understand and evaluate what the level of risk is if a misclassification occurs prior to implementing a solution.

References

Chollet, F., & Safari, an O'Reilly Media Company. (2017). *Deep Learning with Python*(1st ed.). Manning Publications.

Solystic. (2018). Address recognition. Retrieved October 25 2018, from https://www.solystic.com/address-recognition

Appendix

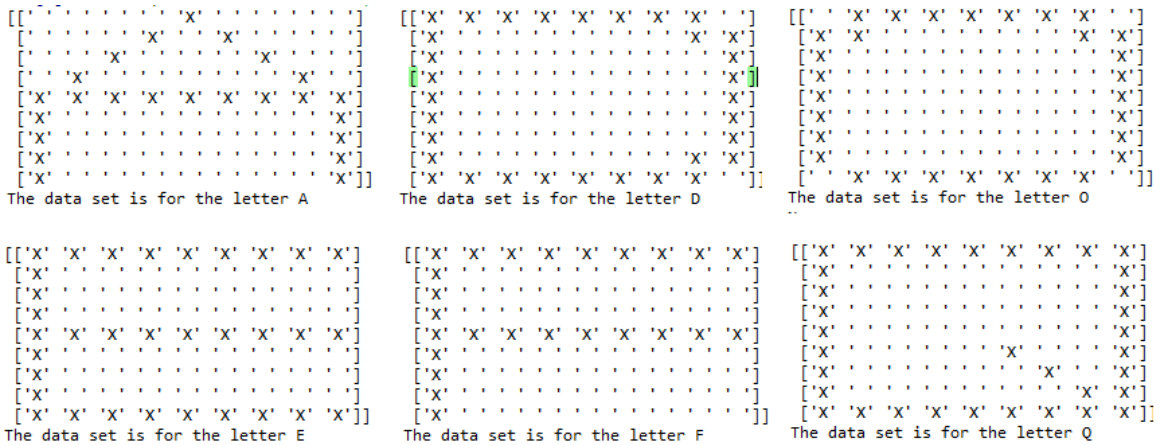

Figure 1 - Plot visualising the arrays used to model individual letters.

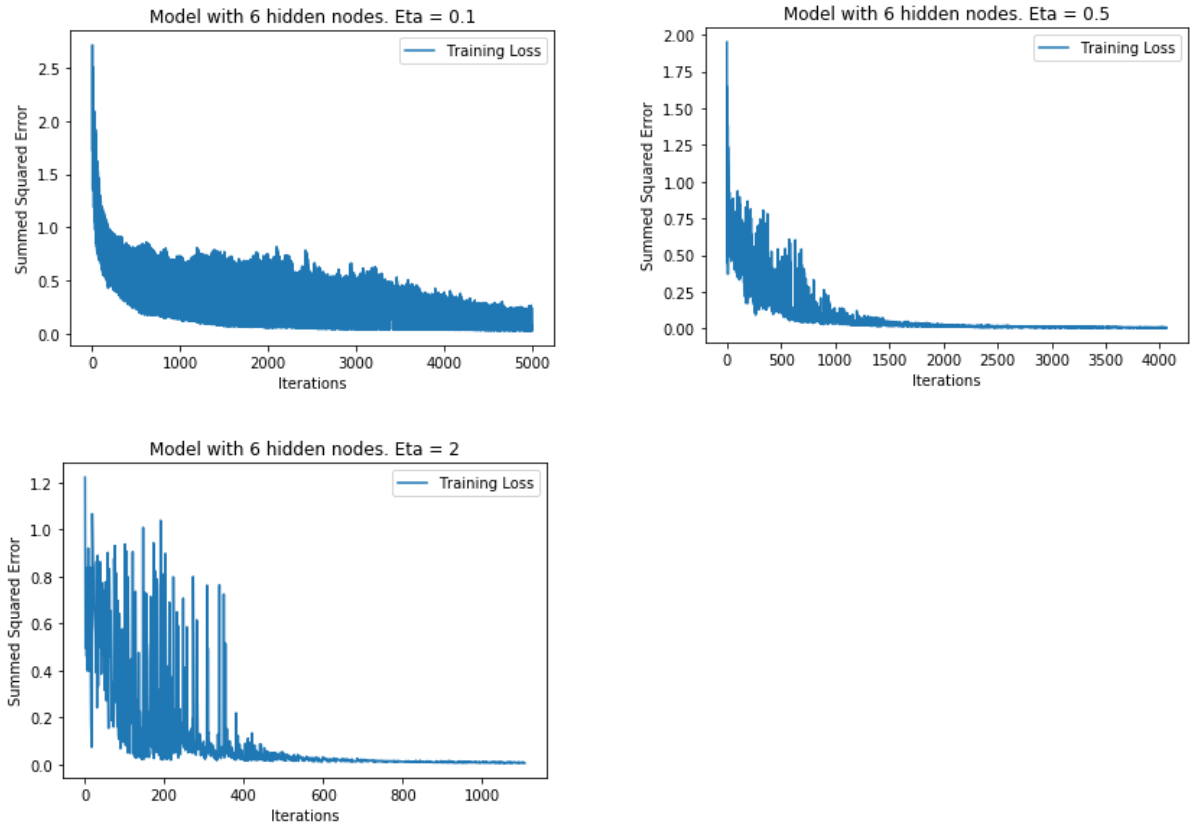

Figure 2 - Plot of loss function vs. iterations with varying learning rates.

Table 1 - Hidden layer node values and output probabilities for a single hidden layer model with 6 hidden nodes.

Hidden layer activation values

Letters	0	1	2	3	4	5
A	0.993595	0.006777	0.181060	0.001514	0.985900	0.038628
D	0.003189	0.925312	0.001786	0.018894	0.000925	0.993442
E	0.062344	0.453642	0.983866	0.871002	0.034696	0.018908
F	0.924886	0.957096	0.997428	0.004702	0.027592	0.915293
O	0.000292	0.009730	0.000929	0.496859	0.002857	0.983170
Q	0.001005	0.990411	0.005435	0.987075	0.003521	0.984811

Output layer probabilities

Letters	0	1	2	3	4	5
A	0.965857	0.019096	0.027356	0.034313	0.039094	0.001736
D	0.009089	0.913633	0.007714	0.057824	0.029323	0.074318
E	0.022032	0.000110	0.947303	0.031884	0.011778	0.041972
F	0.022643	0.040056	0.016911	0.939719	0.000363	0.001005
O	0.021584	0.053845	0.006398	0.002941	0.920257	0.063960
Q	0.002220	0.054023	0.040875	0.001171	0.049607	0.912812

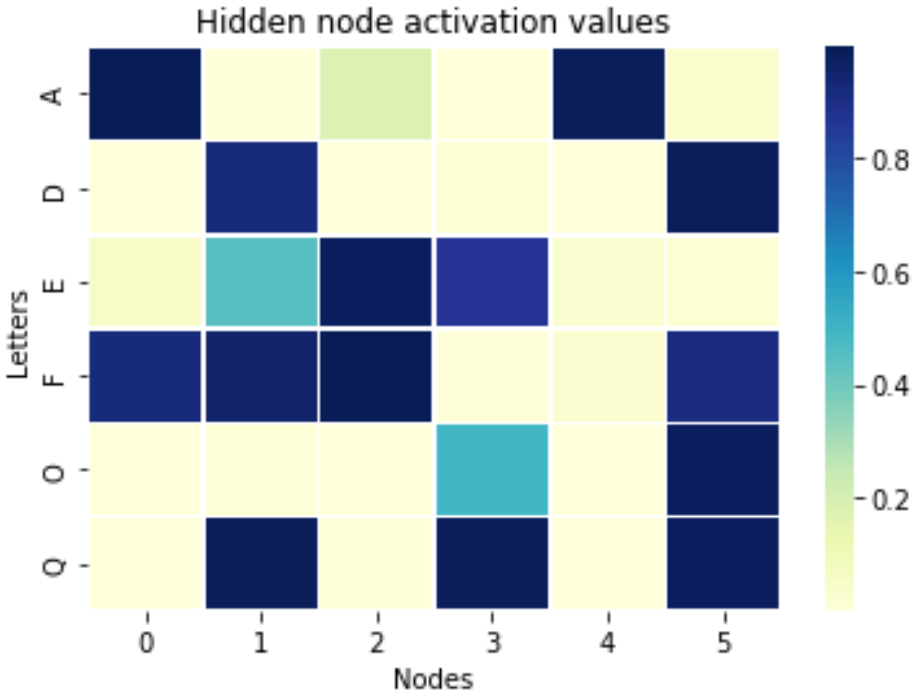

Figure 3 - Activation value heat map for a single hidden layer model with 6 hidden nodes.

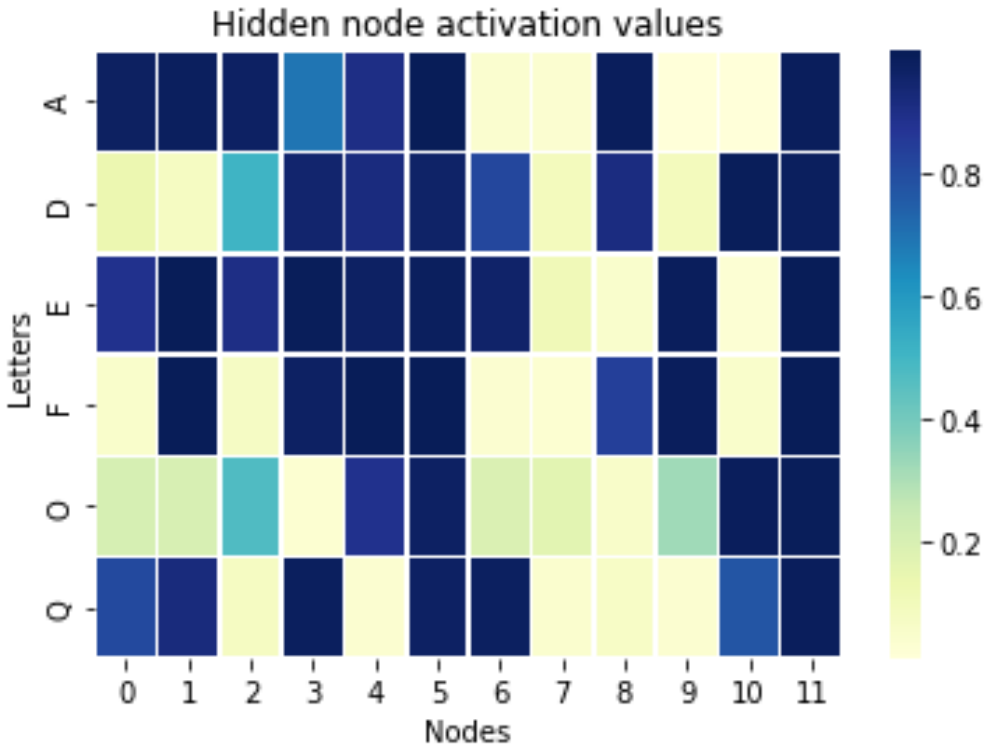

Figure 4 - Activation value heat map for a single hidden layer model with 12 hidden nodes.

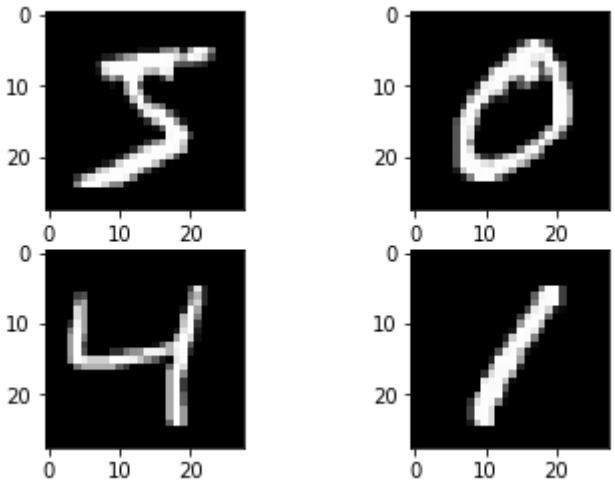

Figure 5 - First four training set images from the MNIST data set.

Table 2 - Frequency distribution of the labels for the training and test data sets.

Frequency distribution for 60,000 training set observations			Frequency distribution for 10,000 test set observations	
0	5923		0	980
1	6742		1	1135
2	5958		2	1032
3	6131		3	1010
4	5842		4	982
5	5421		5	892
6	5918		6	958
7	6265		7	1028
8	5851		8	974
9	5948		9	1008

Table 3 - Comparison of overall performance and computational time for each network architecture.

Number of hidden layers	Nodes per hidden Layer	Computational Time	Training Set Accuracy	Test Set Accuracy
1	100	22sec	0.9915	0.9764
1	250	39sec	0.9985	0.9798
1	500	67sec	0.9993	0.9826
2	250	49	0.9989	0.9810

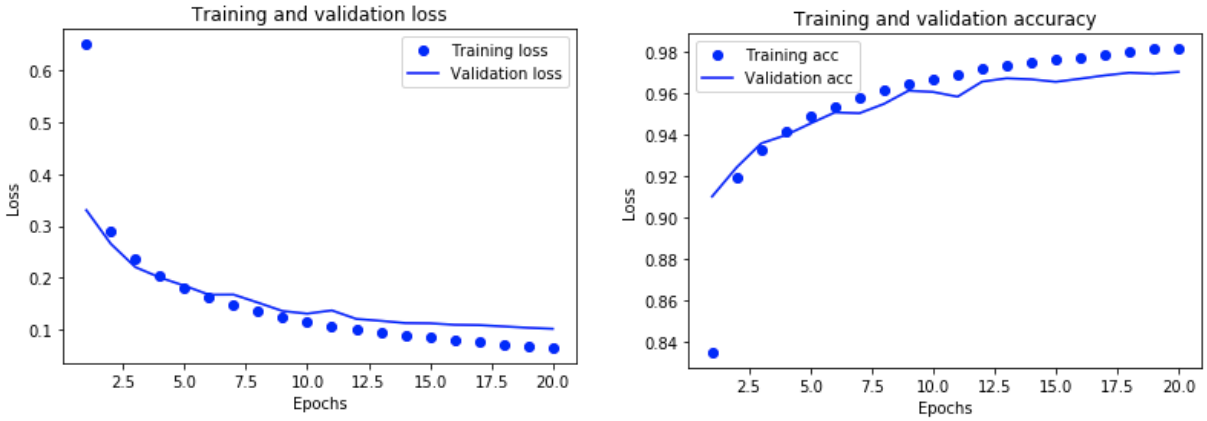

Figure 6 - Training and validation performance for model with single hidden layer with 100 nodes.

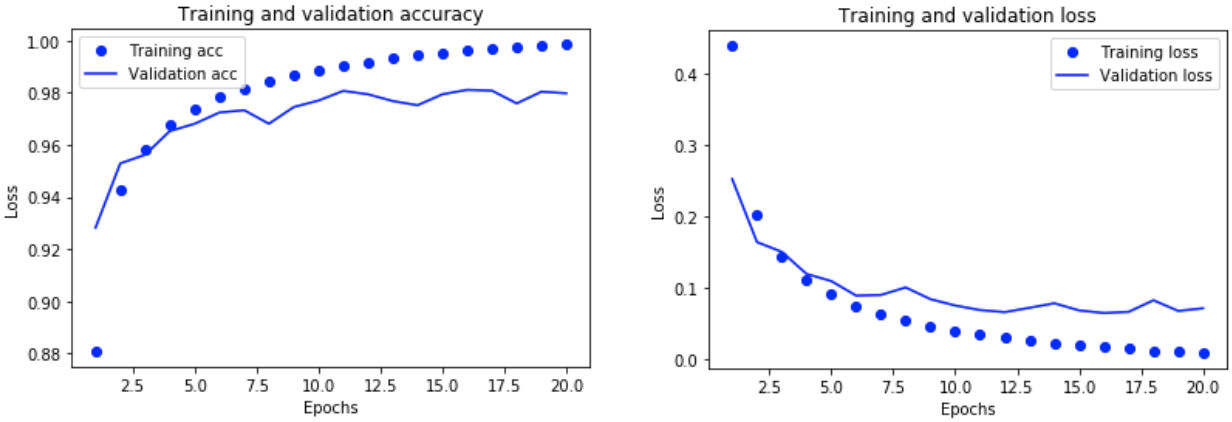

Figure 7 - Training and validation performance for model with single hidden layer with 250 nodes.

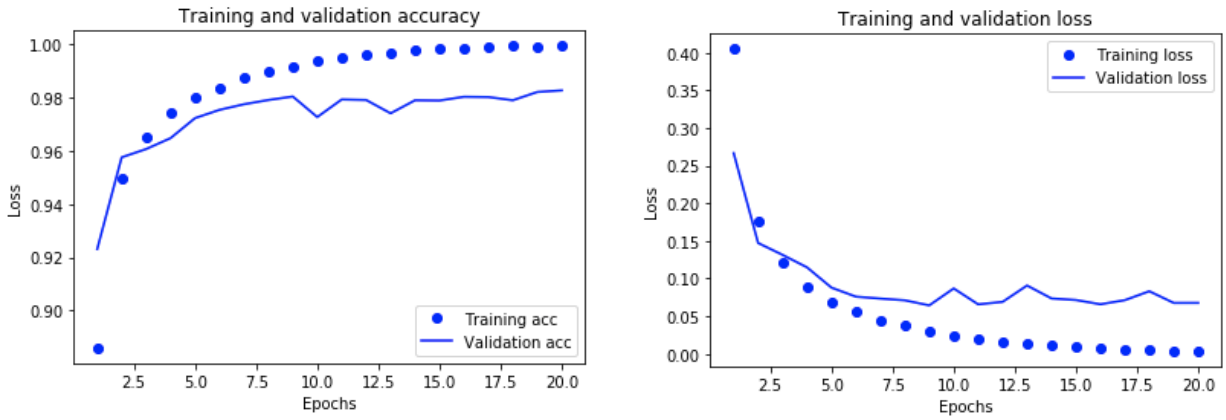

Figure 8 - Training and validation performance for model with single hidden layer with 500 nodes.

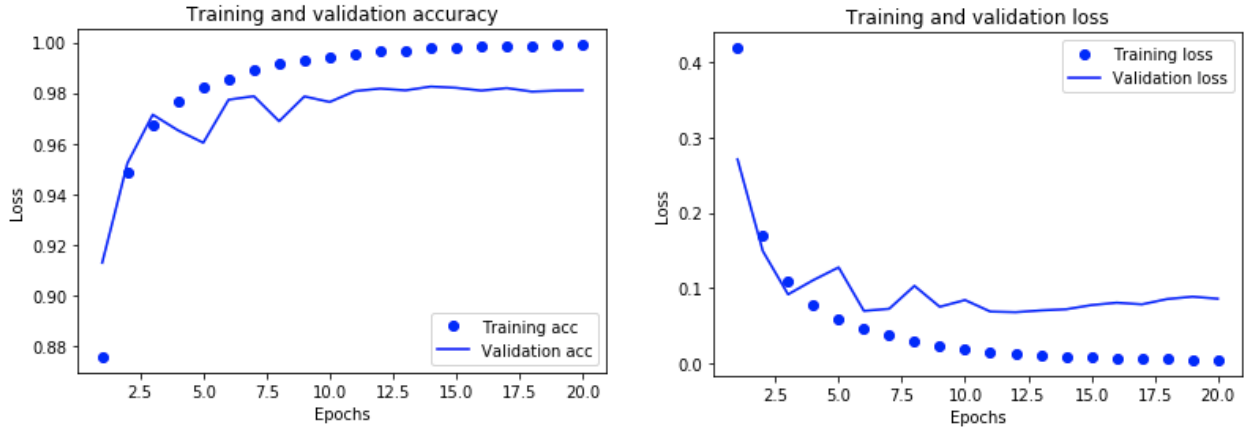

Figure 9 - Training and validation performance for model with two hidden layers each with 250 nodes.