Deep Learning in Computer Vision Lab 3: Neural Networks - Implementing a Neural Network with Keras

15/10/2019

Tara Vanhatalo
M2 Modélisation Mathématique pour le Signal et l'Image
Université de Bordeaux

Overview

The goal of this lab is to create a neural network containing at first only a single neuron, then a neural network with one hidden layer & a binary output and finally a neural network with one hidden layer and an output comprised of ten neurons. To do this, we will be using the MNIST dataset that consists of 70000 grayscale images of handwritten numbers of size 28 by 28.

The first 60000 images will be used to train our model and the other 10000 will be used to test it. To do this, we'll be using the Keras library for Python.

We'll be testing various batch sizes, numbers of neurons on the hidden layer, activation functions and optimizers. Firstly we'll be using the sigmoid activation function with the cross entropy loss calculation and using the stochastic gradient descent as our optimizer.

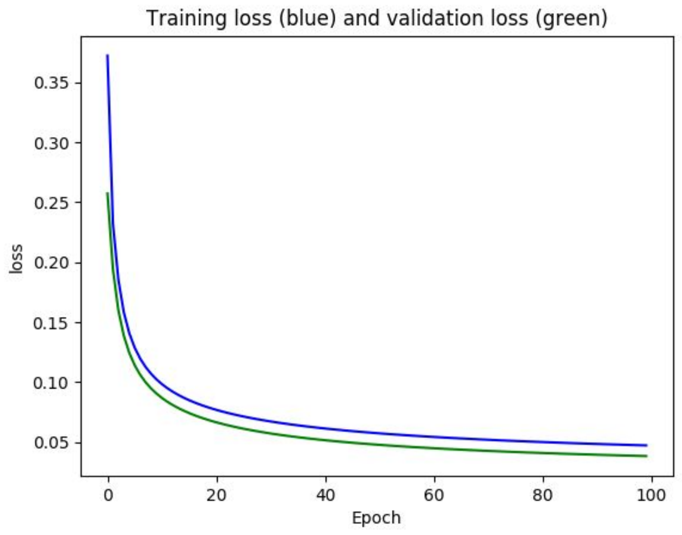
For the models with binary outputs, we'll be using the binary cross entropy whereas for the model with an output of ten neurons we'll be using categorical cross entropy.

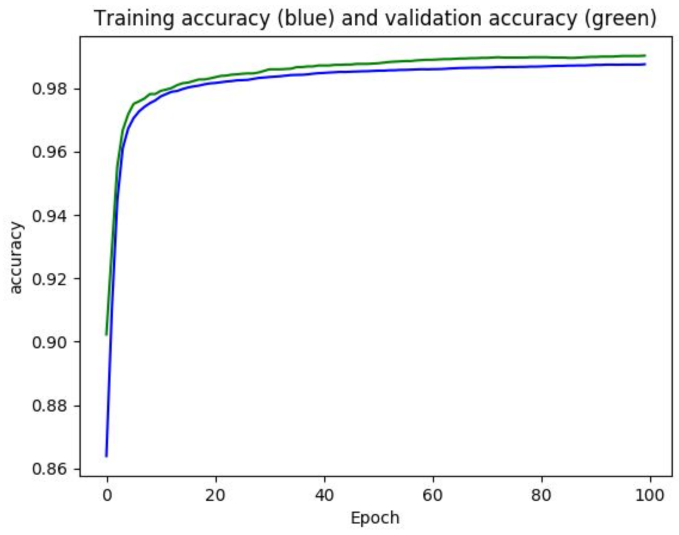
We iterate for 100 epochs.

A Single Neuron

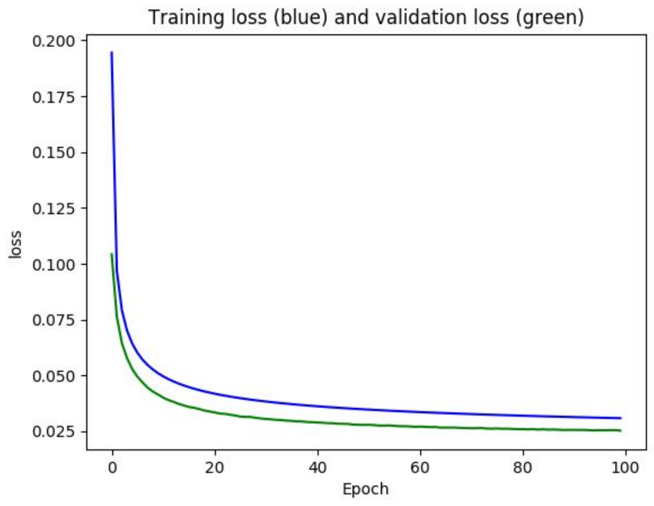
All tests were made on a model with the sigmoid activation function, the stochastic gradient descent optimizer and binary cross entropy for the loss function.

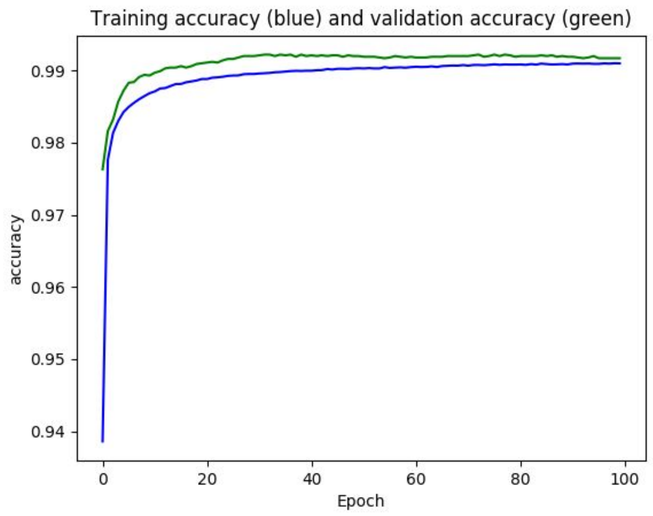
Firstly, we chose a batch size of 1000 and obtained these figures:



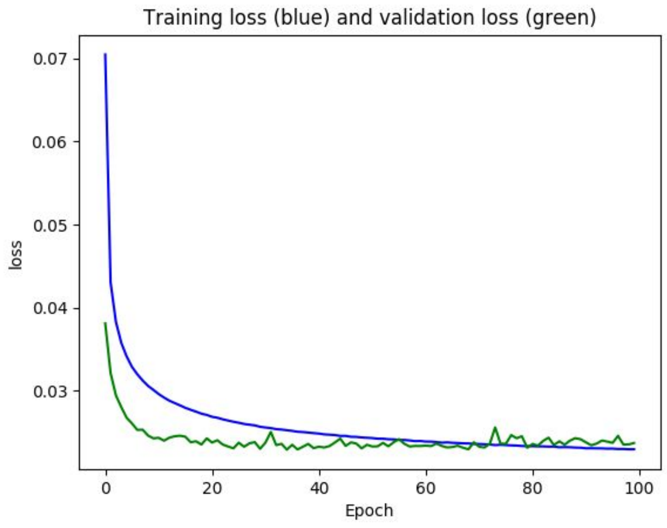


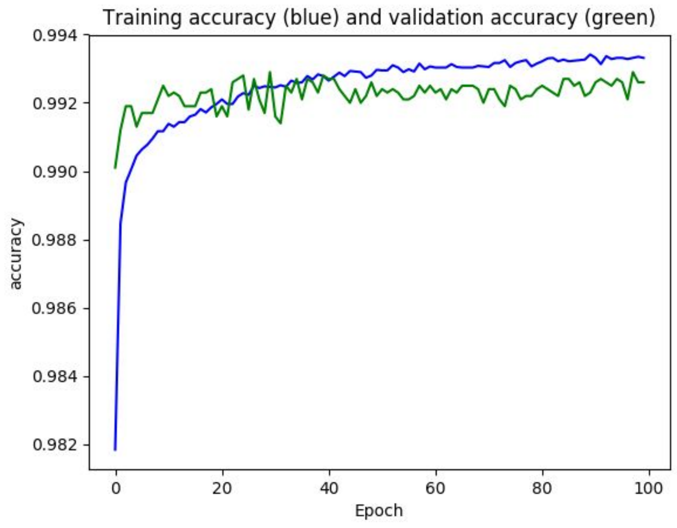
We then used a batch size of 128:





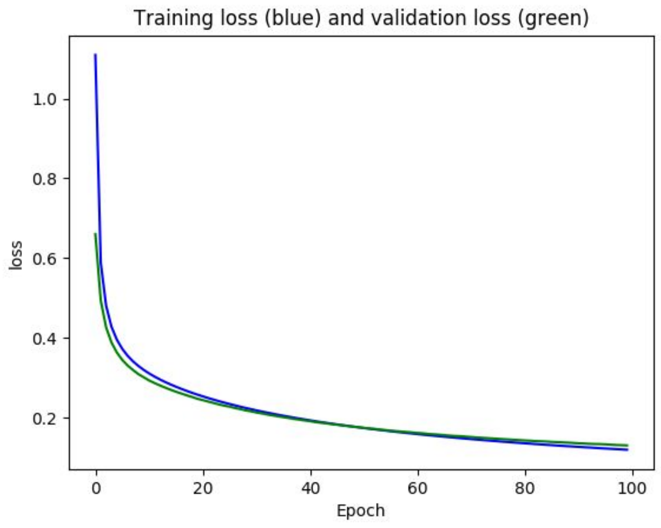
Then we tried a batch size value of 10:

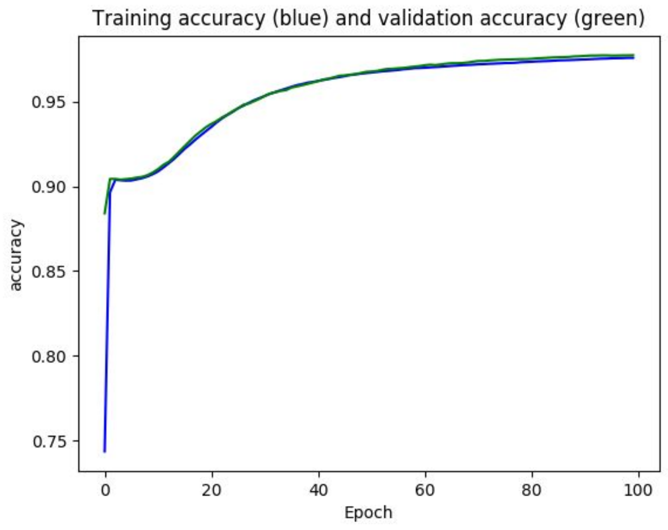




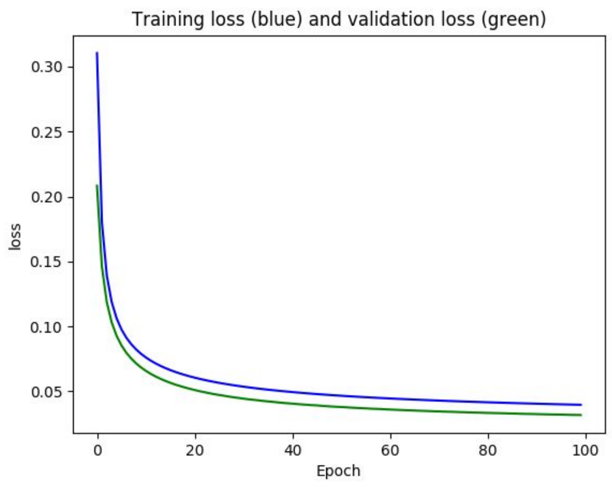
We noticed that the batch size greatly influences the computational speed of our model. The smaller the batch size, the longer it takes to compute for one epoch. Also, the batch size has an effect on our accuracy also. We noticed that a batch size of 10 is a lot less accurate on our test set than one equal to 1000 and the most accurate here is the batch size at 128. So we tested two more batch sizes.

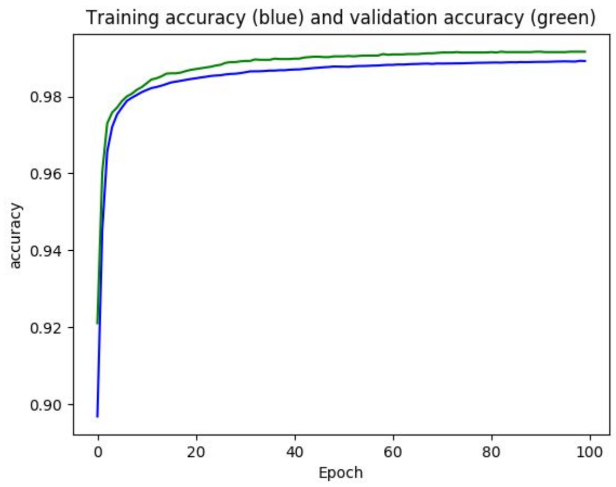
First, batch_size=10000:





And finally, batch_size=512:





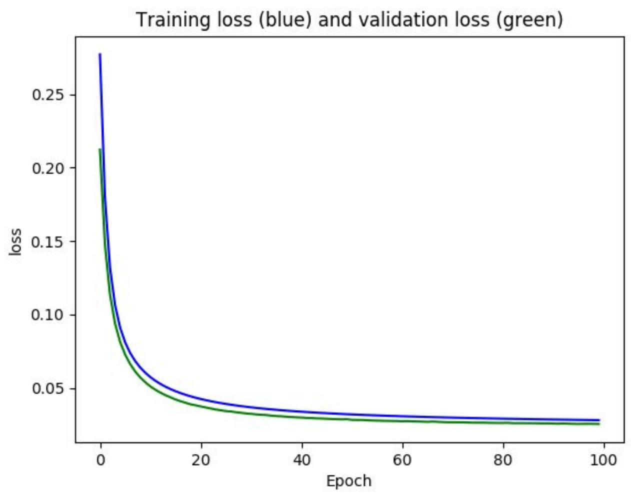
When the batch size is over 1000, the accuracy decreases. The optimal batch sizes seem to be around 128-512. For the rest of the lab we will therefore be using a batch size of 128.

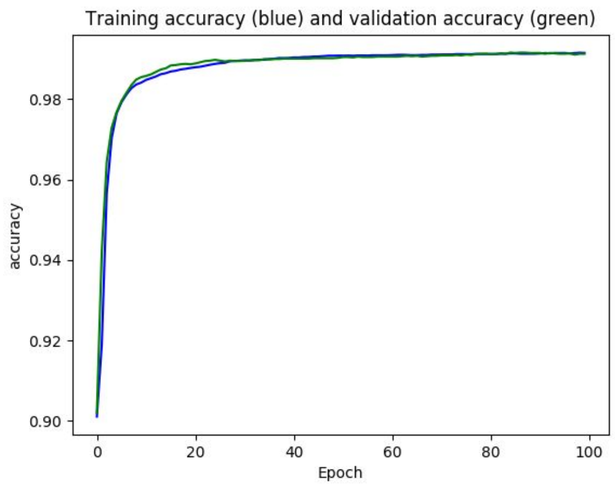
A Neural Network with one Hidden Layer

For a hidden layer of 64 neurons

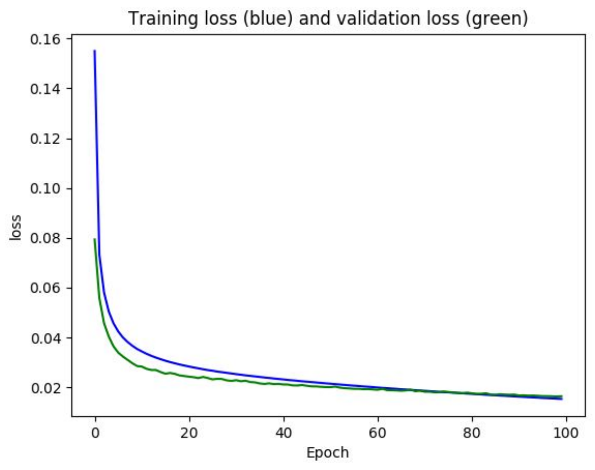
For each model tested, we'll keep using the sigmoid activation function on the last layer.

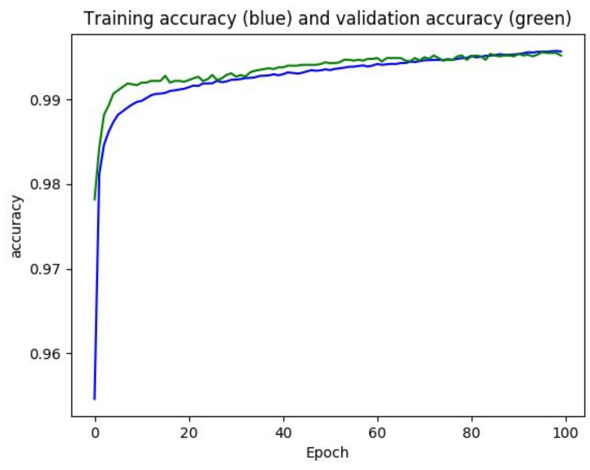
Firstly, we implement a model using the sigmoid activation function, the sgd optimizer still, and a hidden layer with 64 neurons.



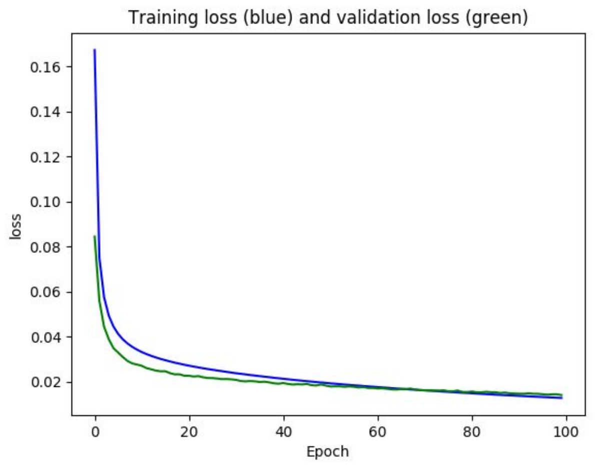


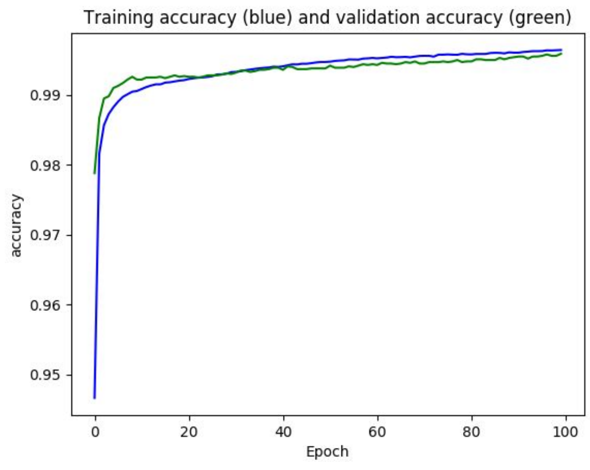
Next, we change our sigmoid activation function to the tanh function, keeping the same optimizer, number of neurons on the hidden layer and number of epochs.





And finally, we try out the relu activation function for a hidden layer of 64 neurons.

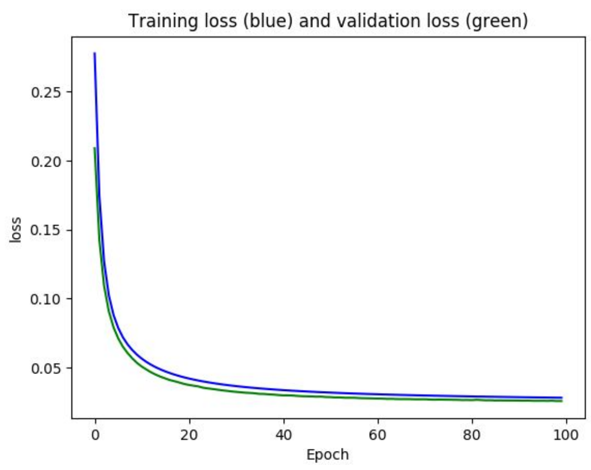


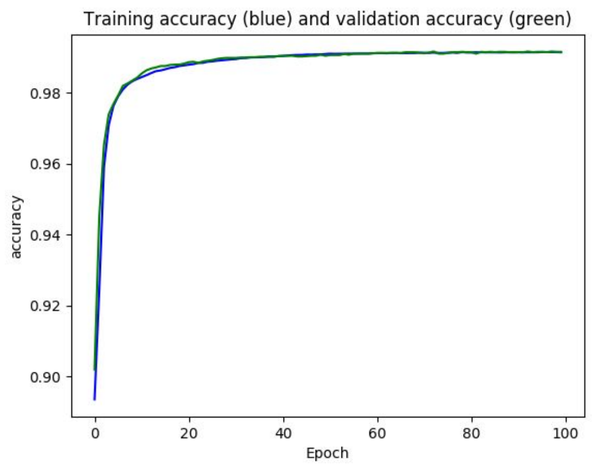


For a layer of 128 neurons

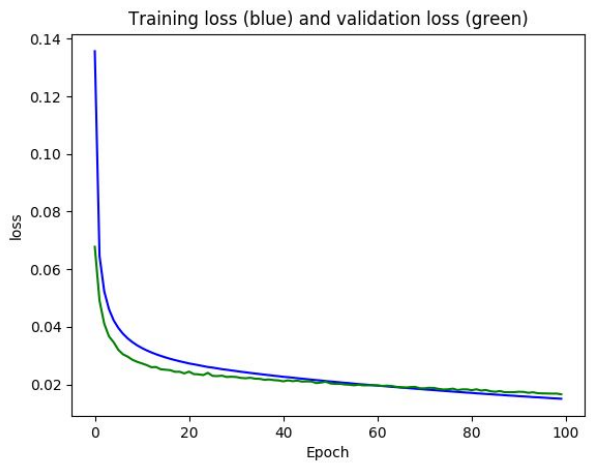
In this section we double our number of neurons on the hidden layer to have 128 neurons, and we test out the same activation functions as previously.

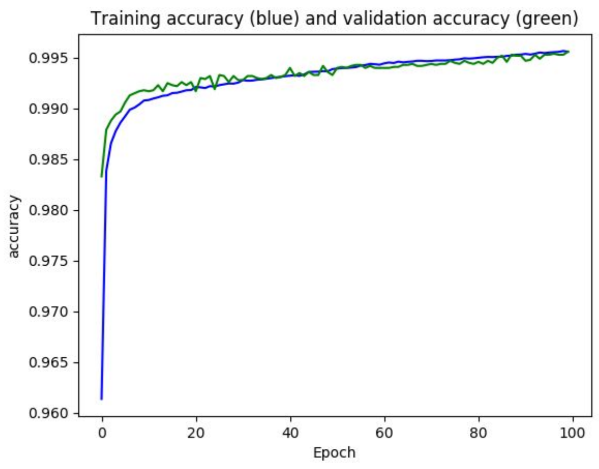
First, the sigmoid function.



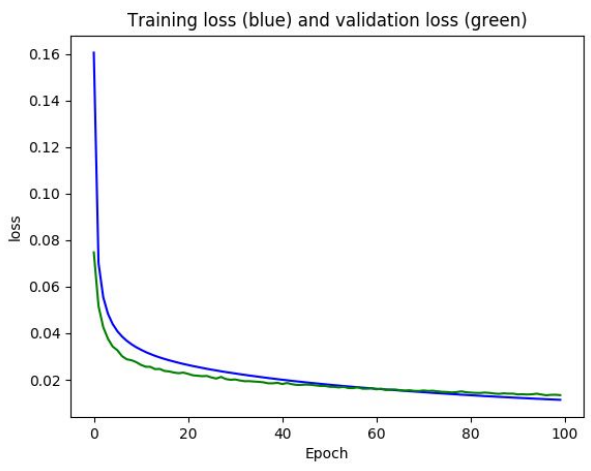


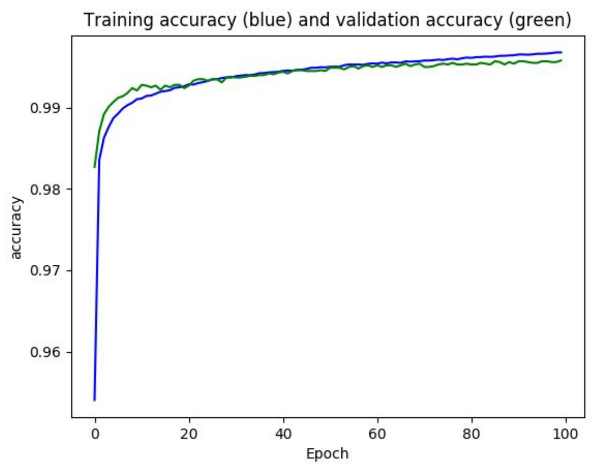
Then the tanh activation function.





And finally the relu function.





Comments

For 128 neurons on the hidden layer we have the following accuracies after 100 epochs:

	Train	Test
Sigmoid	0.9914	0.9915
Tanh	0.9956	0.9956
Relu	0.9968	0.9958

We can therefore assume that the activation function that provides the most accurate results is the relu function.

So we tested this relu activation function on various hidden layer sizes.

	2	64	500	5000
Train	0.9930	0.9968	0.9973	0.9976
Test	0.9923	0.9956	0.9963	0.9966

Therefore we can assume that the accuracy increases with respect to the number of neurons on the hidden layer. However, above 128 neurons, the increase is quite small.

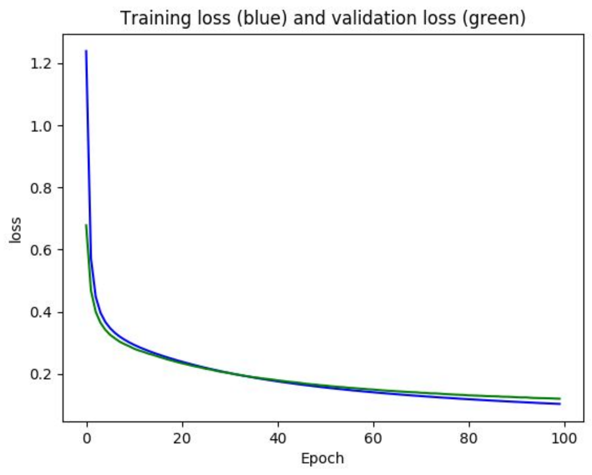
We'll now put in place a neural network with a non-binary output, that is it doesn't just distuinguish between zeros and other digits but can differentiate between all different digits (we will therefore have an output layer with 10 neurons). To do this we use keras to implement "one hot encoding".

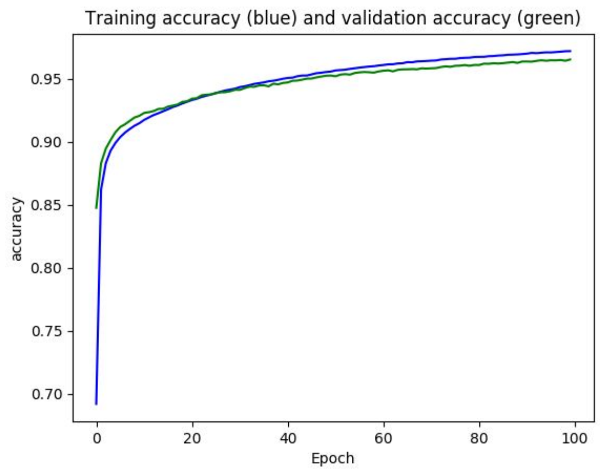
A Multiclass Neural Network

For this part of the lab, we switched for binary cross entropy for the loss calculation to categorical cross entropy to account for the non-binary output. We also changed our activation function for the last layer to the 'softmax' function.

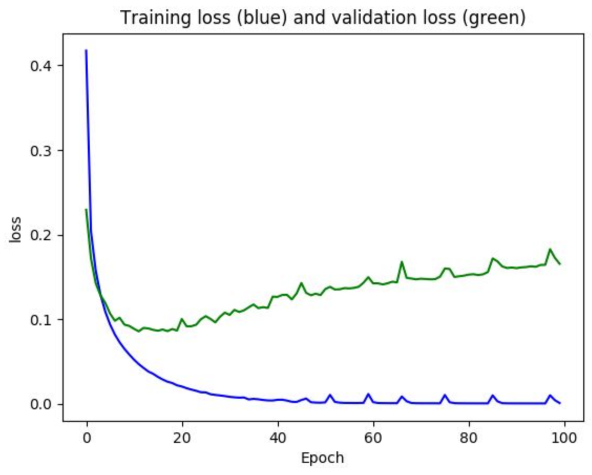
We will first start by trying different optimizers for our model in an effort to increase accuracy. Since the relu activation function appeared to be the most accurate in the last part we will start with this function with 64 neurons on the hidden layer.

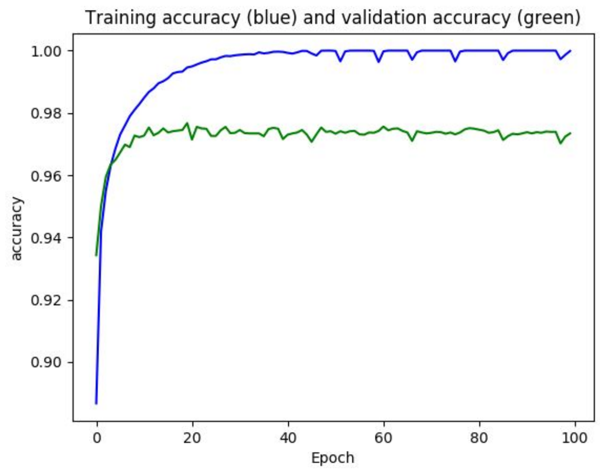
Relu paired with the stochastic gradient descent optimizer:



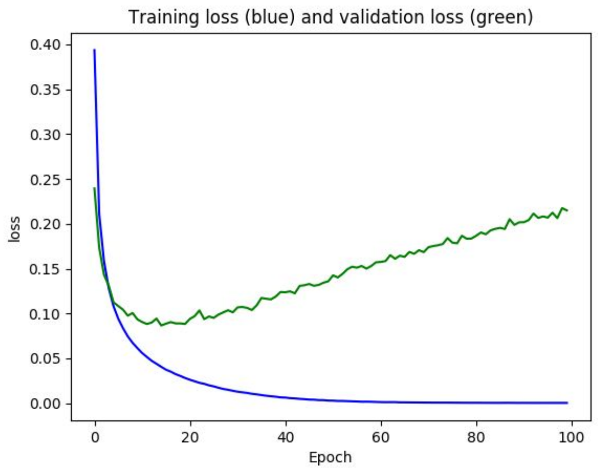


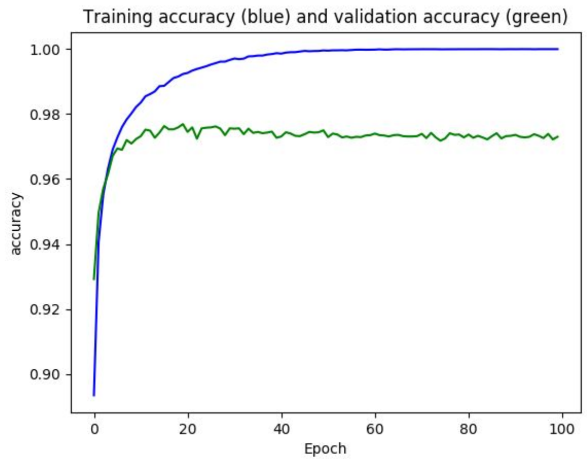
Relu paired with the adam optimizer:





And finally, relu with the RMSprop optimizer:





Comments

Here are the accuracies for our optimizers:

	Train	Test
SGD	0.9720	0.9654
Adam	0.9999	0.9734
RMSprop	1.0000	0.9730

The most accurate results are obtained with the Adam and RMSProp optimizers. However, the accuracy of the model on the test set is a lot lower than on the train set for these two optimizers, and when we observe the plots of the loss function on these two models we clearly have overfitting. Therefore, our best model here would be with the stochastic gradient descent optimizer.

Conclusion: The Most Accurate Model

We saw in the first part of this lab that the optimal batch size was around 128 to 512 so to have the most accurate model possible we should use for example a batch size of 128. We then found the most optimal activation function to be the relu function and the best optimizer for it to be paired with was the stochastic gradient descent.