Report Practical assignment 1

Machinists:
Fares Ben Slimane

February 6, 2019

Abstract

This report explain our approaches to solving the problems of the practical assignment 1. the experiments we performed, results and conclusion of our work.

1 Problem 1 (MLP)

1.1 Building model

1. (using the python script mlp.py under the folder problem1), we build an MLP with two hidden layers h1(512 hidden units) and h2(1024 hidden units).The total number of parameters of the network $= 784 * 512 + 512 + 512 * 1024 + 1024 + 1024 * 10 + 10 = 937,482 \approx 0.9M$.

2. We Implemented the forward and backward propagation in a generalized way (can work in any number of layers) of the MLP in numpy without the use of a deep learning framework and using the provided class structure. (See python script mlp.py under problem 1 folder).

3. We trained the MLP using the probability loss (cross entropy) as training criterion (See loss method in the NN class) and minimize this criterion to optimize the model parameters using stochastic gradient descent (See the update method in the NN class). (See python script mlp.py under problem 1 folder).

1.2 Initialization

We consider a model architecture of two hidden layers h1 = 24 hidden units and h2 = 12 hidden units and a total number of parameters of 19270. We chose RELU as an activation function, a learning rate of 0.01 and a mini-batch size of 1000.

1. We trained the model for 10 epochs using the 3 initialization methods (Zero, normal and glorot) and we recorded the average loss measured for each method.

    - Zero initialization: 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3, 2.3
    - Normal initialization: 3.41, 2.25, 2.20, 2.18, 2.16, 2.15, 2.13, 2.11, 2.08, 2.03

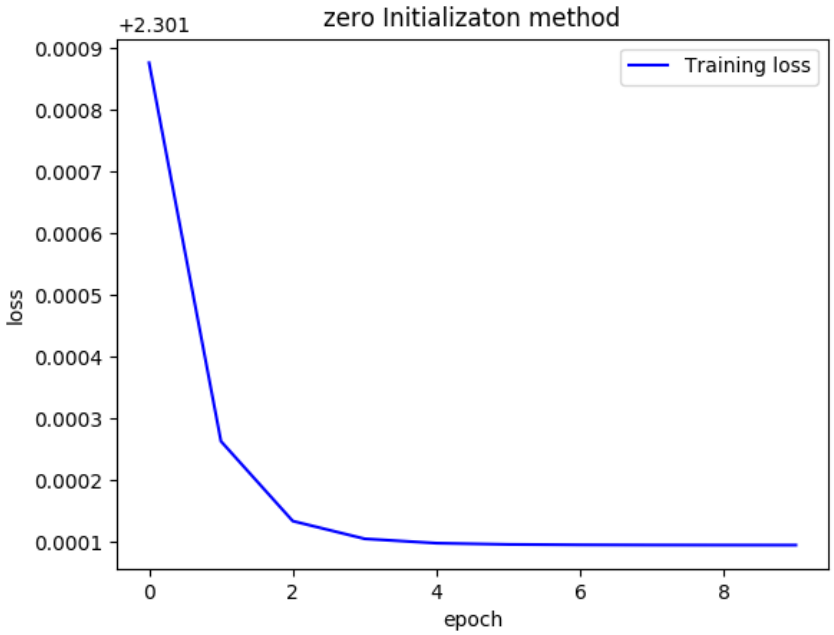

Figure 1: average loss against the training time (epoch) using zero initialization method.

- Glorot initialization: 1.55, 0.55, 0.40, 0.35, 0.32, 0.30, 0.29, 0.27, 0.26, 0.25

2. We plot the losses against the training time (epoch) using each initialization method (Figures 1, 2 and 3). We conclude from the plots that the glorot initialization is the best among the methods in which the loss decreases rapidly at each epoch whereas, for the zero initialization, the loss decreases very very slowly. An explanation for this is that by initializing all the weights to zero. All the hidden nodes will end up with the same value and therefore we end up learning just one function. This is called the symmetry problem. We break the symmetry problem by initializing the weights randomly (like we did in the glorot and normal initializations).

1.3 Hyperparameter Search

1. The combination of hyper-parameters that we found in which the average accuracy rate on the validation set reach 97.2% accuracy: A network with 2 hidden layers h1 = 64 hidden units and h2 = 32 hidden units with a total number of parameters of 52650. A Relu activation function, a learning rate of 0.01, a mini batch size of 64, a number of epochs of 50. (See Figure **??** plotting the training/validation accuracy over epochs).

2. We tried different hyper-parameters for learning rate, number of epochs,

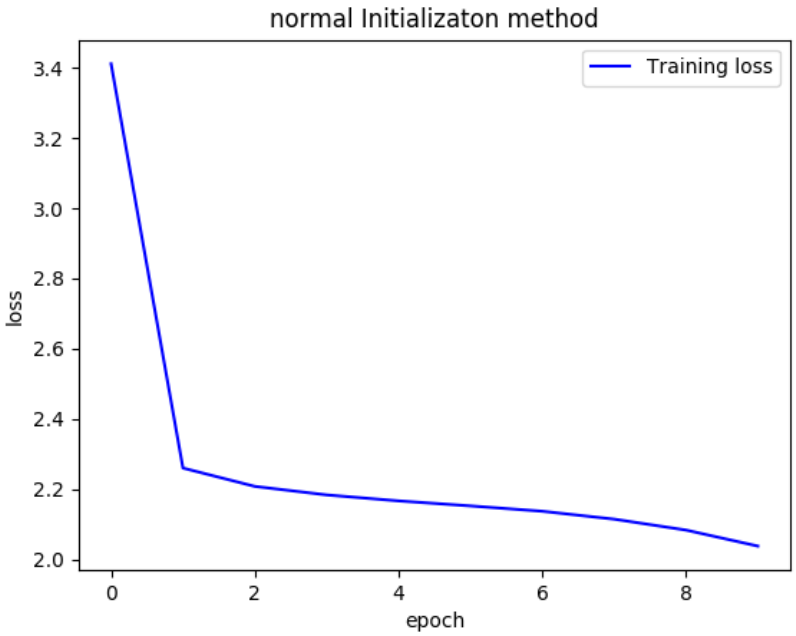

Figure 2: average loss against the training time (epoch) using normal initialization method.

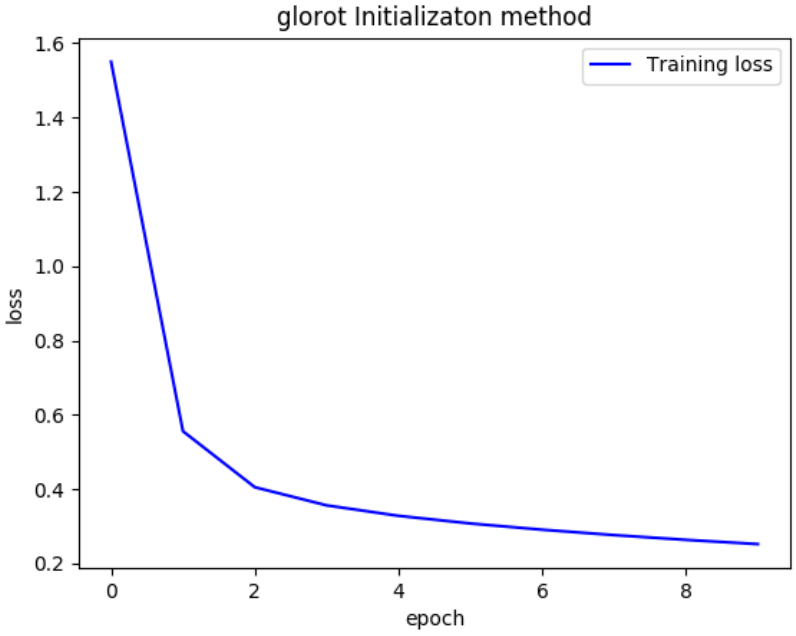

Figure 3: average loss against the training time (epoch) using glorot initialization method.

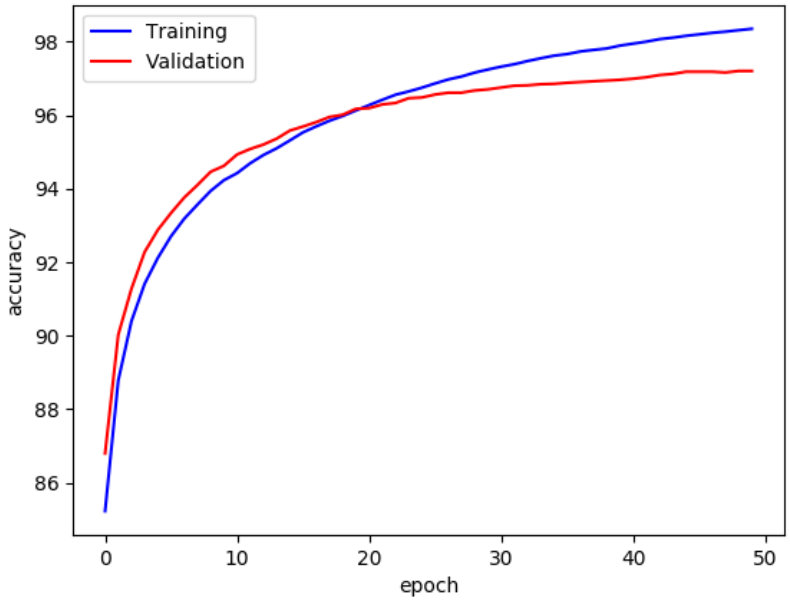

Figure 4: Validation/training accuracy against the training time (epoch) using the chosen hyper-parameters.

network architecture and batch size. We consider the same model architecture.

1.3.1 Learning rate

We keep the same settings and we only change the learning rate.

- learning rate of 0.01: validation accuracy = 97.2% (See Figure 4).
- learning rate of 0.1: validation accuracy = 97.6% (See Figure 5).
- learning rate of 0.001: validation accuracy = 93% (See Figure 6).
- learning rate of 0.5: validation accuracy = 29.8% (See Figure 7).

1.3.2 Number of epochs

We keep the same settings and we only change the number of epochs.

- number of epochs of 50: validation accuracy = 97.2% (See Figure 4).
- number of epochs of 5: validation accuracy = 92.8% (See Figure 8).
- number of epochs of 150: validation accuracy = 97.3% (See Figure 9).

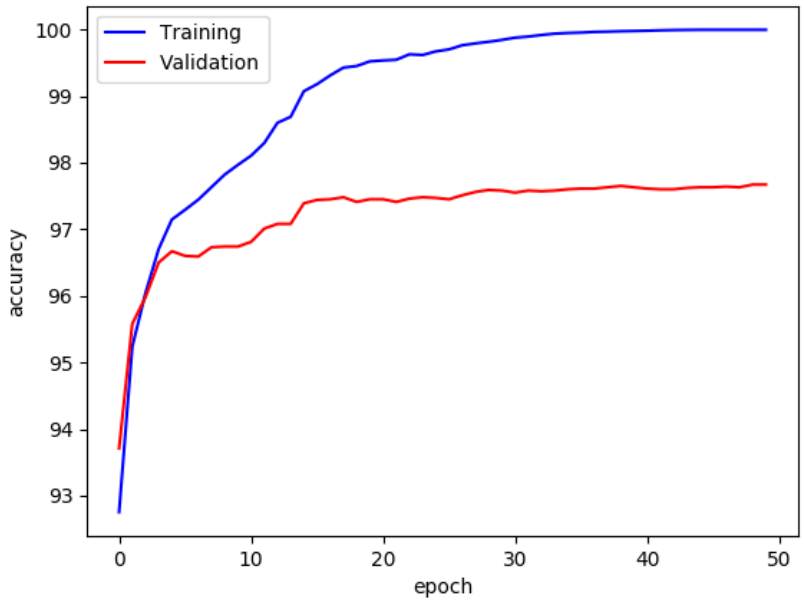

Figure 5: Validation accuracy against the training time (epoch) using a learning rate of 0.1.

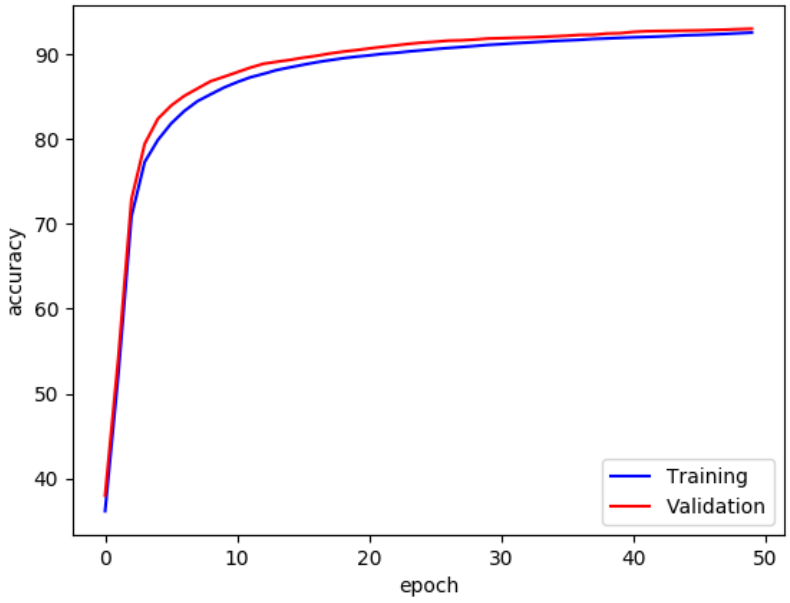

Figure 6: Validation accuracy against the training time (epoch) using a learning rate of 0.001.

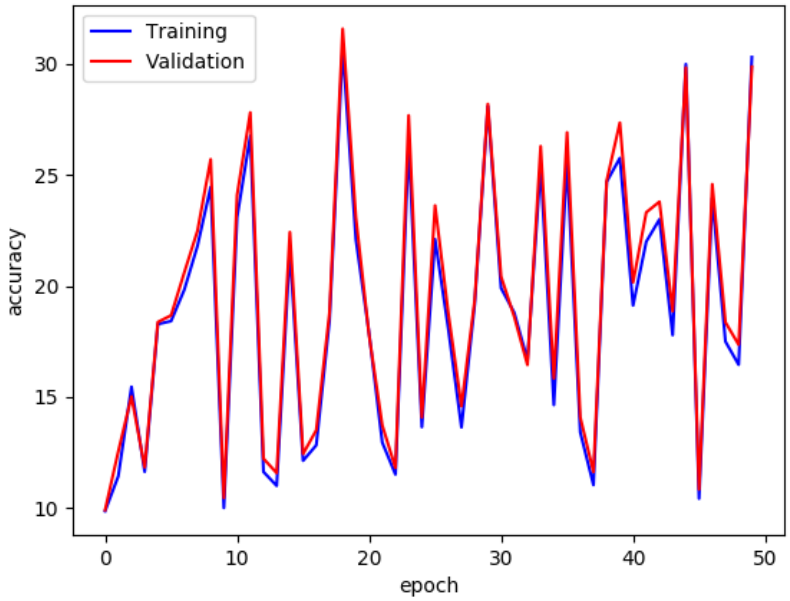

Figure 7: Validation accuracy against the training time (epoch) using a learning rate of 0.5.

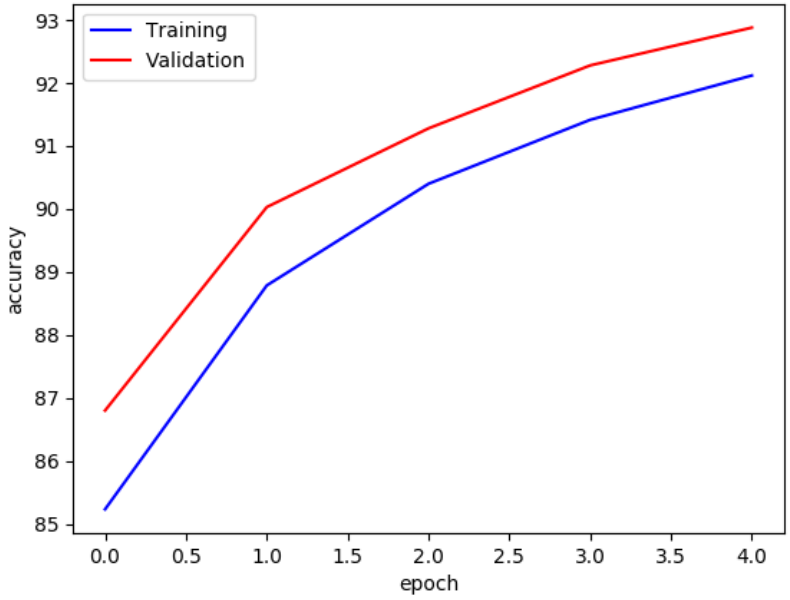

Figure 8: Validation/training accuracy against the training time (epoch) using a number of epochs of 5.

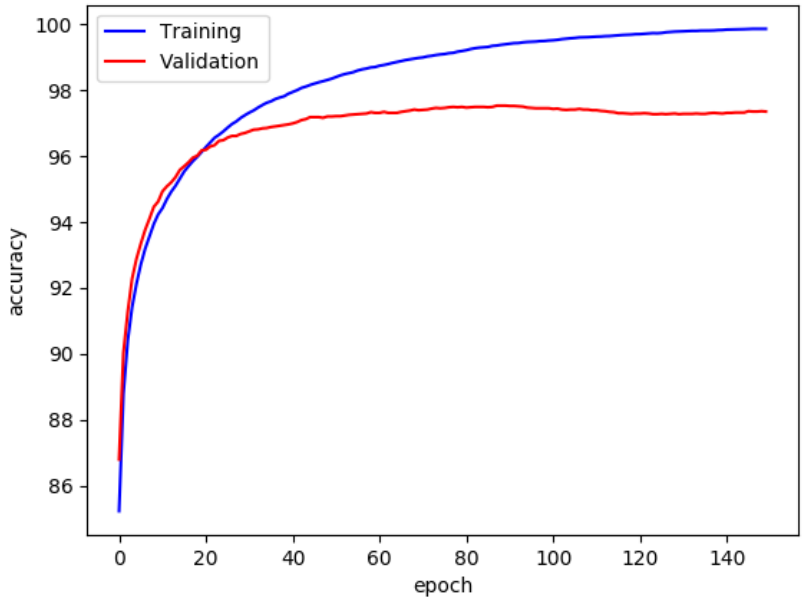

Figure 9: Validation/training accuracy against the training time (epoch) using a number of epochs of 150.

1.3.3 Number of mini-batches

We keep the same settings and we only change the number of mini-batches.

- number of mini-batches of 64: validation accuracy = 97.2% (See Figure 4).
- number of mini-batches of 8: validation accuracy = 97.4% (See Figure 10).
- number of mini-batches of 512: validation accuracy = 93.4% (See Figure 11).

1.3.4 Non-linearity activation function

We keep the same settings and we only change the activation function.

- Relu activation function: validation accuracy = 97.2% (See Figure 4).
- Sigmoid activation function: validation accuracy = 92.2% (See Figure 12).
- tanh activation function: validation accuracy = 93.9% (See Figure 13).

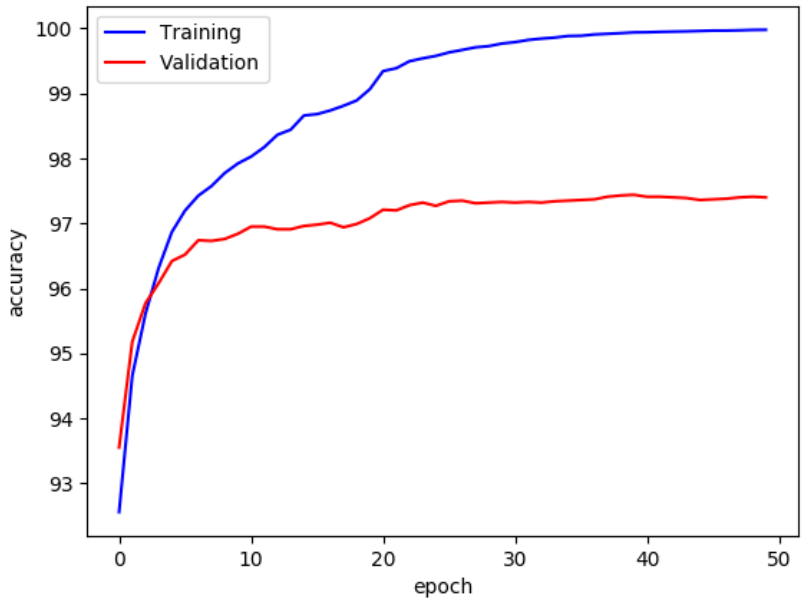

Figure 10: Validation/training accuracy against the training time (epoch) using a number of mini-batches of 8.

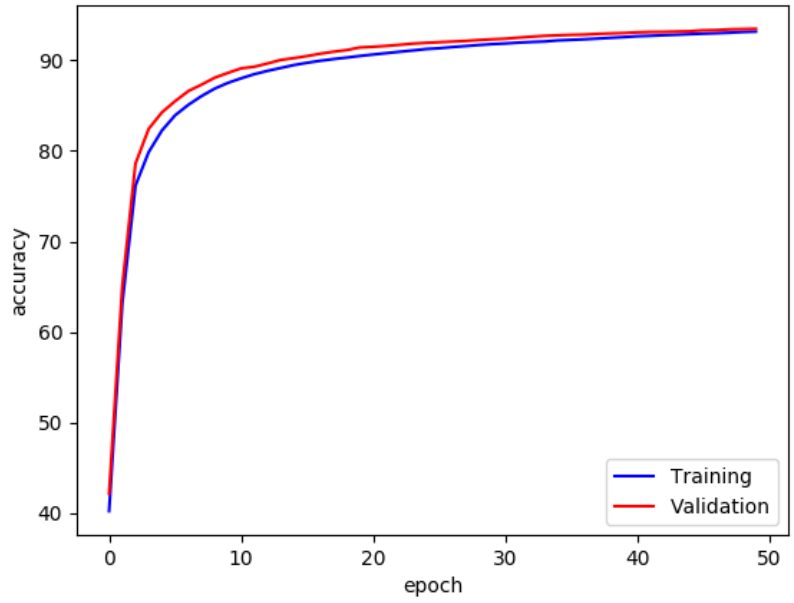

Figure 11: Validation/training accuracy against the training time (epoch) using a number of mini-batches of 512.

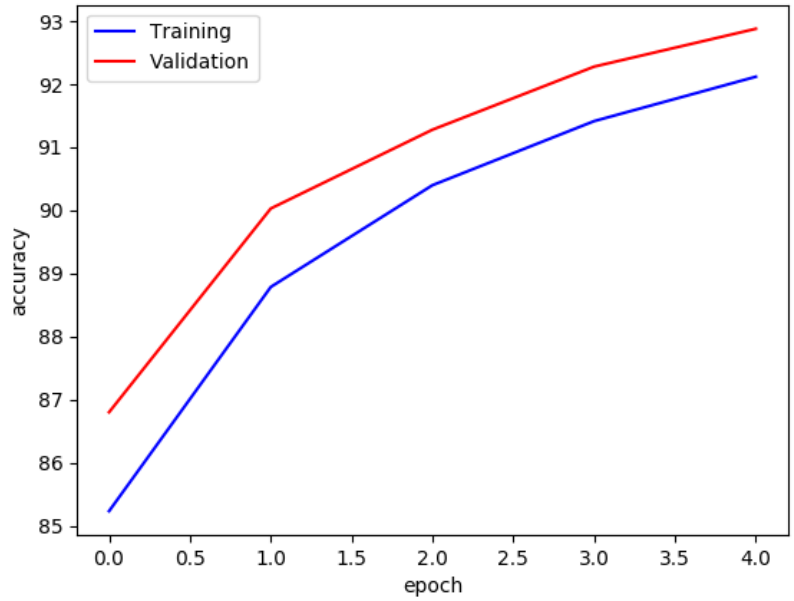

Figure 12: Validation/training accuracy against the training time (epoch) using a sigmoid activation function.

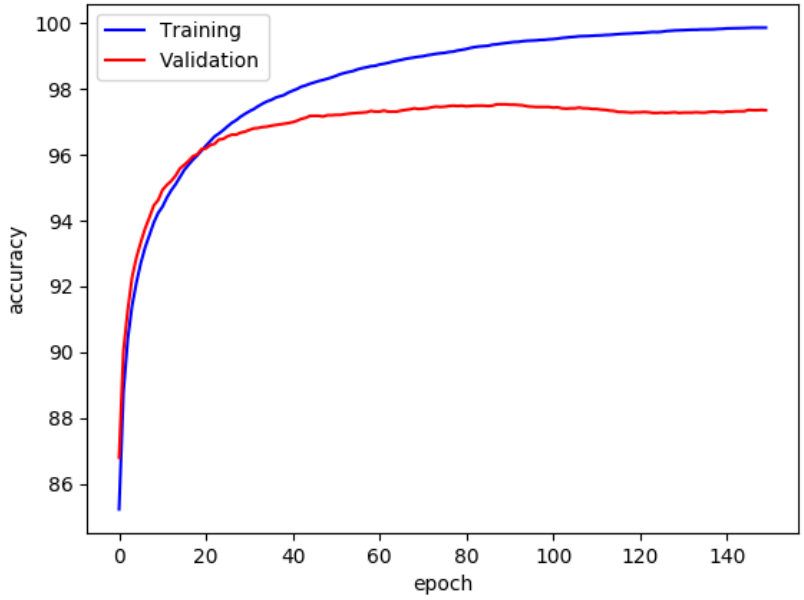

Figure 13: Validation/training accuracy against the training time (epoch) using a tanh activation function.

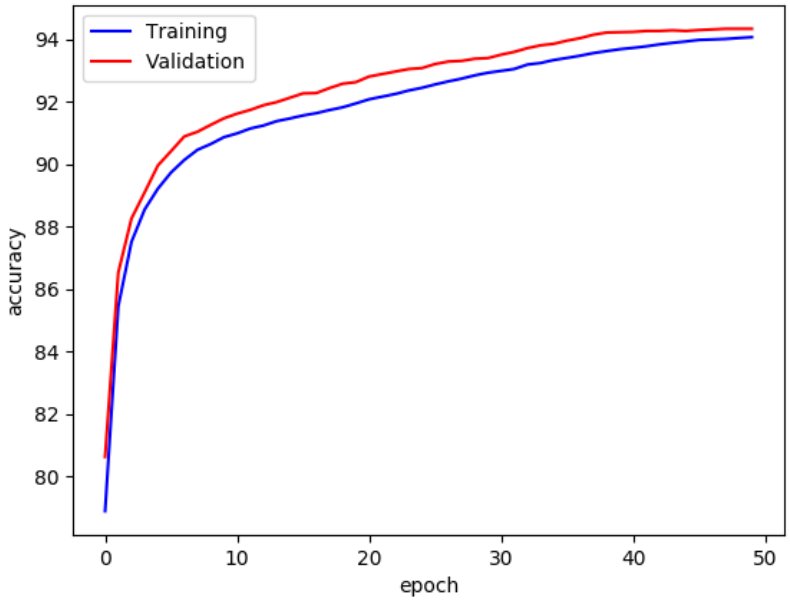

Figure 14: Validation/training accuracy against the training time (epoch) using a network architecture of (h1=32, h2=64).

1.3.5 Network architecture (number of hidden units)

We keep the same settings and we only change the hidden layers dimensions.

- Network with 2 hidden layers h1 = 64, h2 = 32: validation accuracy = 97.2% (See Figure 4).
- Network with 2 hidden layers h1 = 32, h2 = 64: validation accuracy = 94.3% (See Figure 5).
- Network with 2 hidden layers h1 = 10, h2 = 5: validation accuracy = 72.1% (See Figure 6).

2 Problem 2

3 Problem 3 (Kaggle challenge)

@online, author = Thayumanav Jayadevan, title = Why don't we initialize the weights of a neural network to zero?, year = 2018, url = https://www.quora.com/Why-dont-we-initialize-the-weights-of-a-neural-network-to-zero, urldate = 2019-02-05

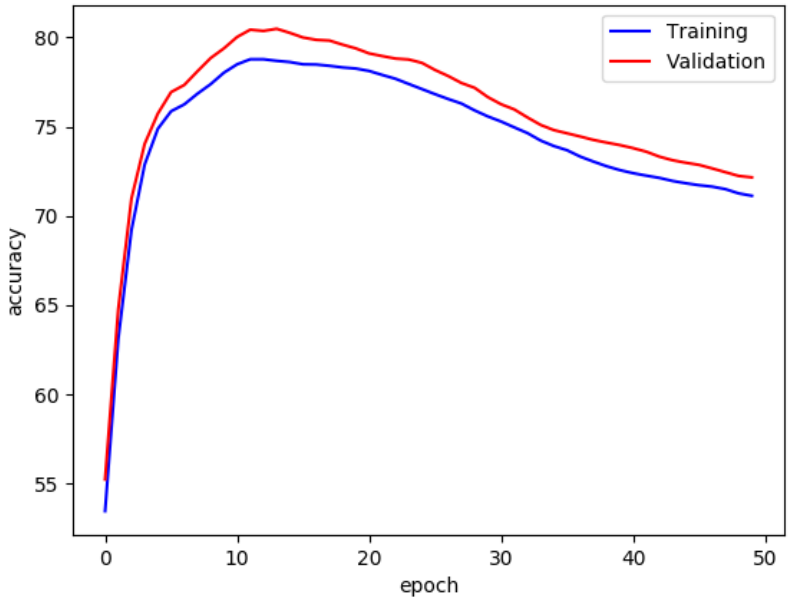

Figure 15: Validation/training accuracy against the training time (epoch) using a network architecture of (h1=10, h2=5).