HW1 – Report

Gal Barak – 204233688

Shahar Stahi – 305237257

Program structure:

assignment_1_code.py: contains the 'main' function. Calls to the functions as requested.

functions.py: Defines and implements the requested functions (except function 1).

generic_module.py: Defines provided the generic module with extensions required for the assignment.

mnist_fasion_module.py: defines a dataset-specific module (Inherits from the generic module)

dataset.py: defines dataset related functionalities (DataLoader and more)

Note: Function 1 was defined and implemented in *dataset.py*. All other functions were defined in *functions.py*

Answers:

1) Models in ascending order:
    a. two_hidden_layers_sigmoid (4) – **0.7241**
    b. two_hidden_layers_relu_SGD_decreasing_lr (4) – **0.8105**
    c. two_hidden_layers_relu (4) – **0.8151**
    d. one_hidden_layer_no_activation (4) – **0.8207**
    e. four_hidden_layers_adam_weight_decay (32) – **0.8219**
    f. two_hidden_layers_relu_adam (4) – **0.8227**
    g. four_hidden_layers_adam_early_stopping (32) – **0.8319**
    h. one_hidden_layer_no_activation (32) – **0.8425**
    i. two_hidden_layers_relu_SGD_decreasing_lr (32) - **0.8583**
    j. two_hidden_layers_sigmoid (32) – **0.8652**
    k. four_hidden_layers_adam (32) – **0.8670**
    l. two_hidden_layers_relu (32) – **0.8729**
    m. two_hidden_layers_relu_adam (32) – **0.8764**
2) Intuitive explanation on the ordering:

    We can see that 32 neurons per layer is usually better than 4 (For this data) and we expect it since the data is very complex and 4 neurons tend to underfit the data.

    On the other side, we can see models that are too complex and have many parameters tend to produce lower than expected results. It happens since they need a lot of data in order to get better predictions and if there's enough of it or we use too many epochs, simpler models can get better results.

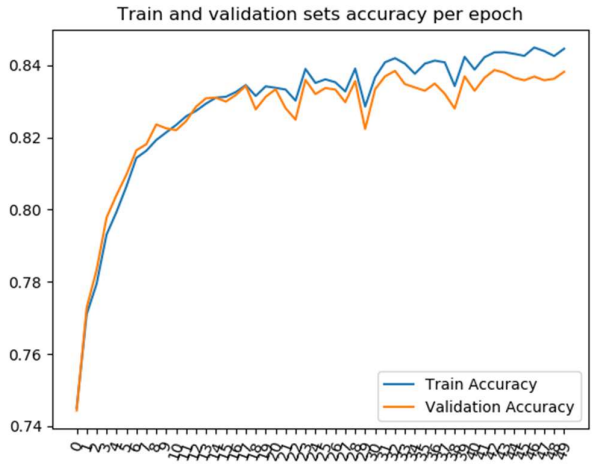
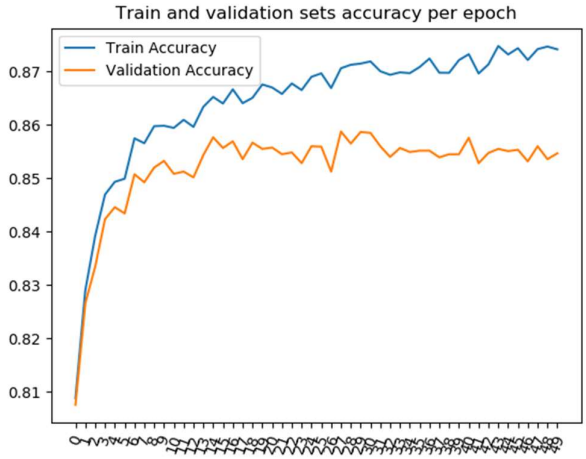
    We can also see that generally, Adam produces better results than SGD and Relu than Sigmoid.

3) Is there any difference between the test and the training accuracy? Why?

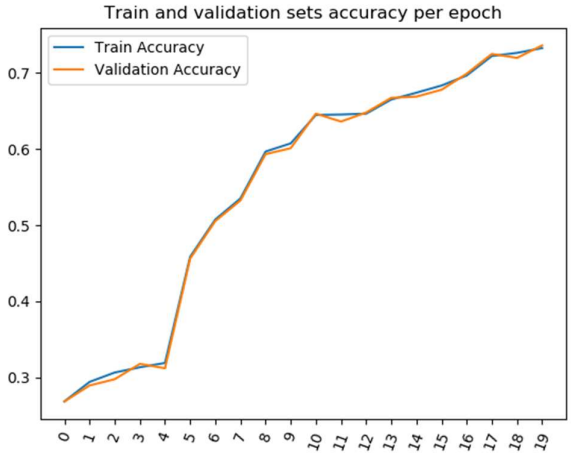
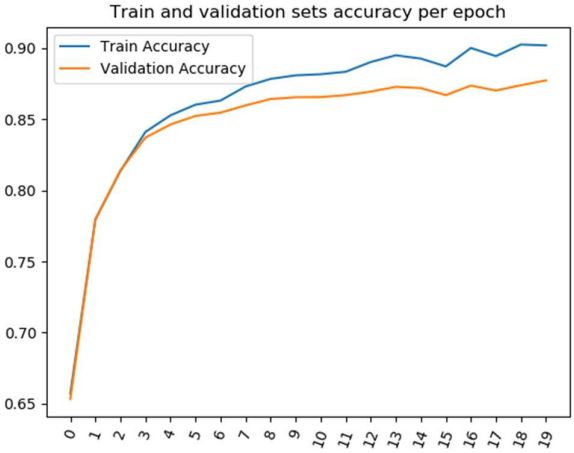
    Generally speaking, the training accuracy is higher than the test accuracy and we expect this to happen. The difference is how big the gap is. In highly complex models we can see a huge gap (100% in training vs 82% in test), while simple-mid complexity produces a small gap and a good estimation of the test.

Plots:

Function 2 - one_hidden_layer_no_activation:

4 neurons:	32 neurons:
	
Test accuracy: 0.8207	0.8425

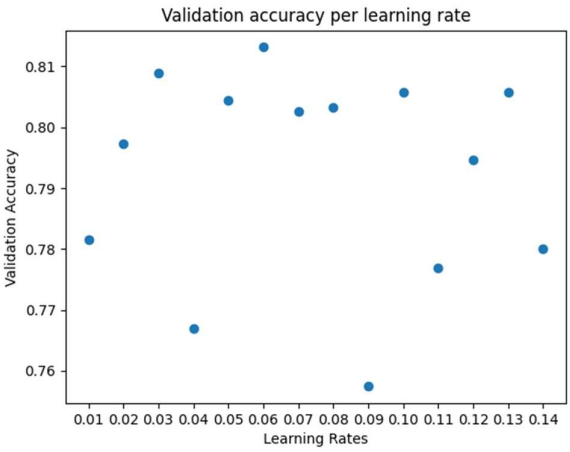
Function 3 - two_hidden_layers_sigmoid:

4 neurons:	32 neurons:
	
Test accuracy: 0.7241	0.8652

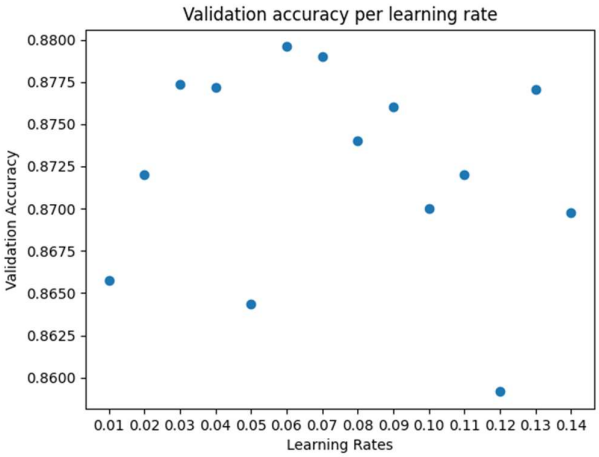
Function 4 - two_hidden_layers_relu:

Validation accuracy per learning rate:

4 neurons:

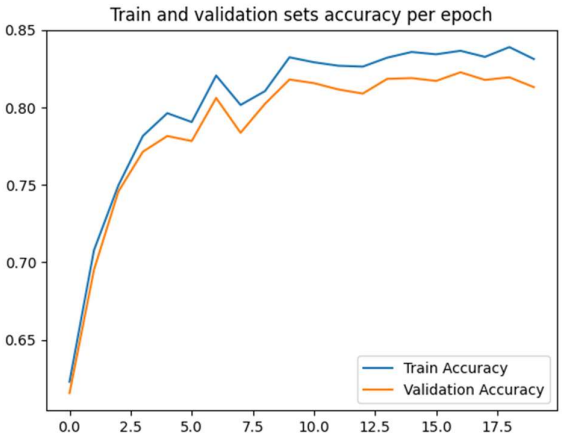


32 neurons:

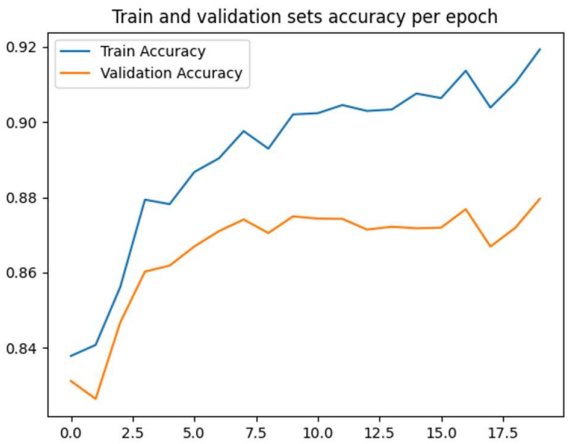


Final model:

4 neurons (lr=0.06):

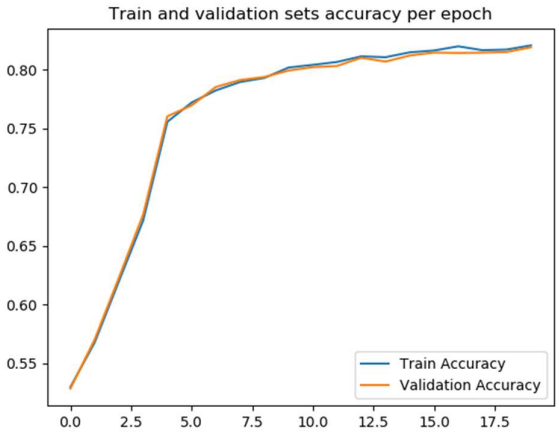
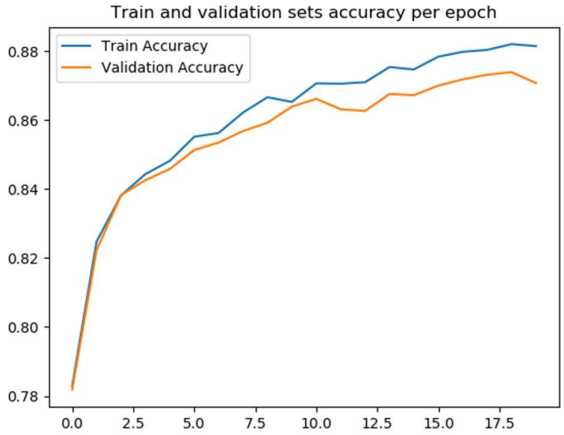


32 neurons (lr=0.06):



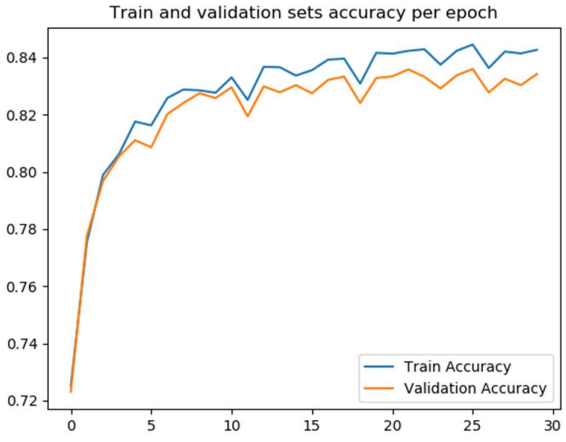
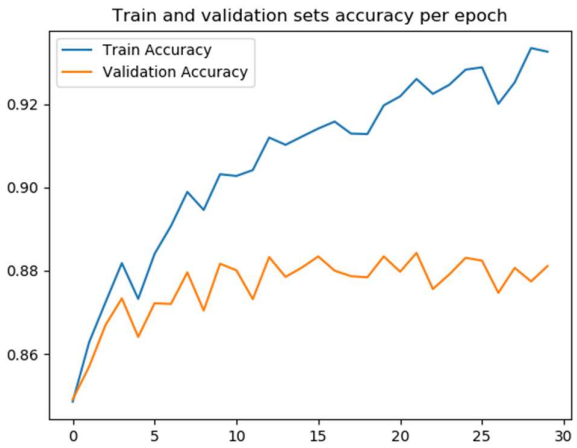
Test accuracy:	0.8151	0.8729

Function 5 - two_hidden_layers_relu_SGD_decreasing_lr:

4 neurons (step_size=3):	32 neurons (step_size=5):
	

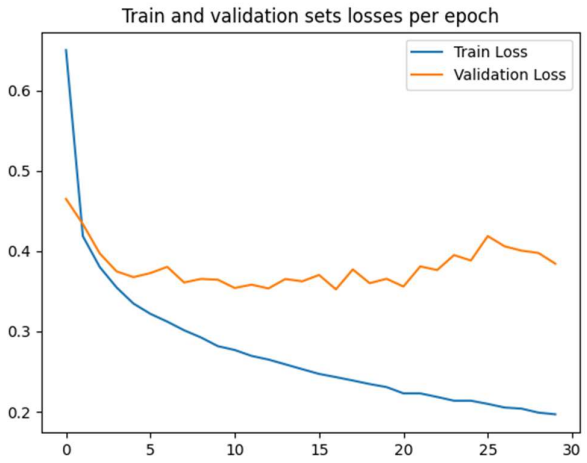
Test accuracy:	0.8105	0.8583

Function 6 - two_hidden_layers_relu_adam:

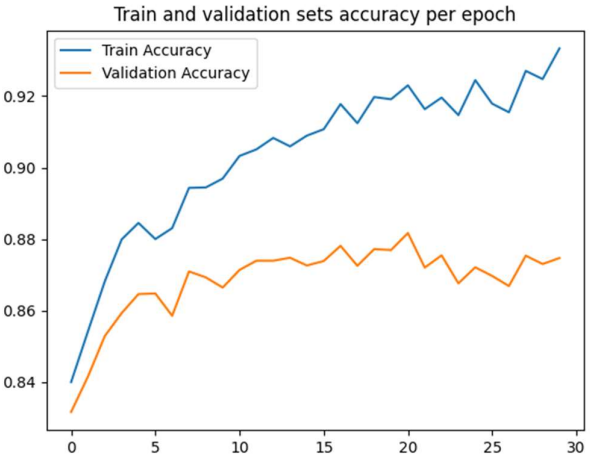
4 neurons:	32 neurons:
	

Test accuracy:	0.8227	0.8764

Function 7 - four_hidden_layers_adam (32 neurons):



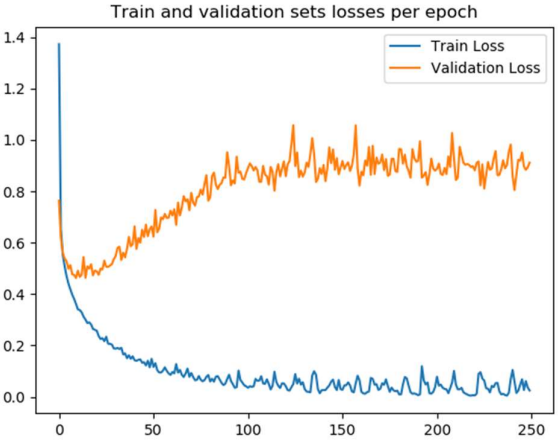
What do we see in this plot?
After around 10 epochs, the training loss continue to decrease but the validation loss doesn't and even increases.
This is a clear sign of **overfitting**, it happens because we continue to train our model based on the training data only (without using the validation loss as some sort of indication to stop).



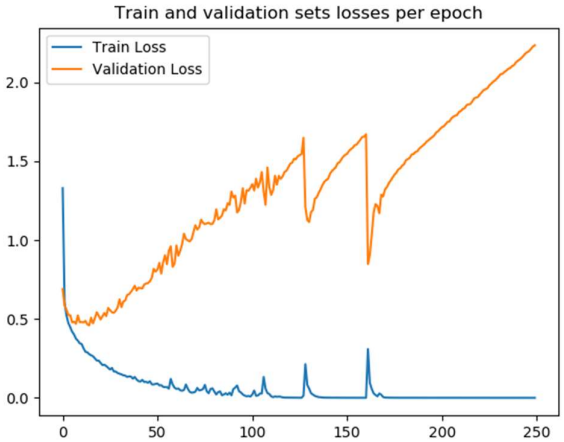
Test accuracy: 0.8670

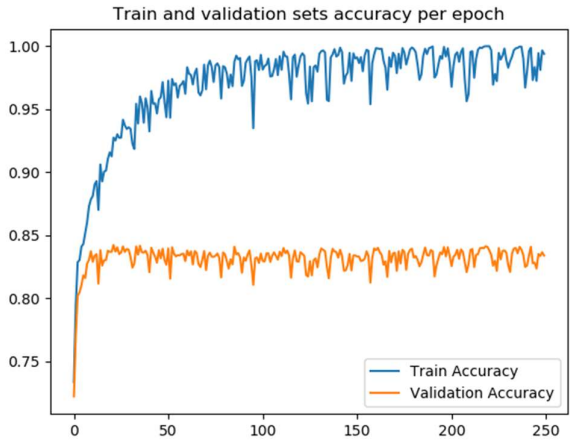
Function 8 - four_hidden_layers_adam_weight_decay (32 neurons):

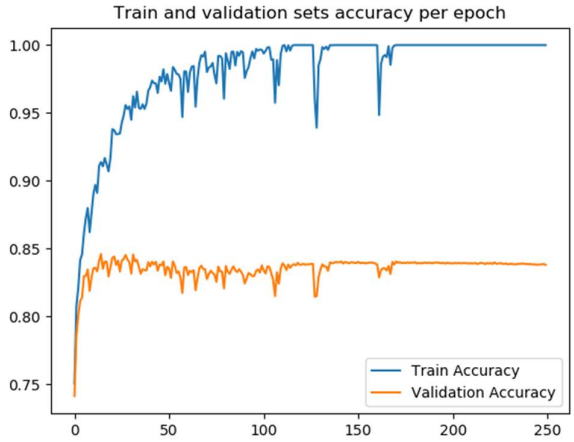
With weight decay:



Without weight decay:







Test accuracy:	0.8219	0.8336

Difference between with/without weight decay:

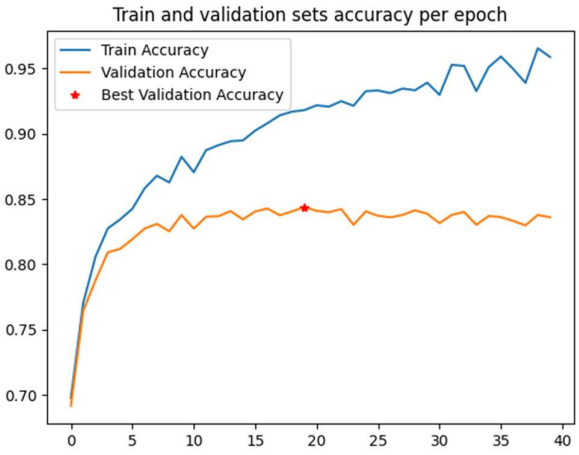
In this function we were asked to use only 10% of the training data as the training set. The rest were used as the validation set.

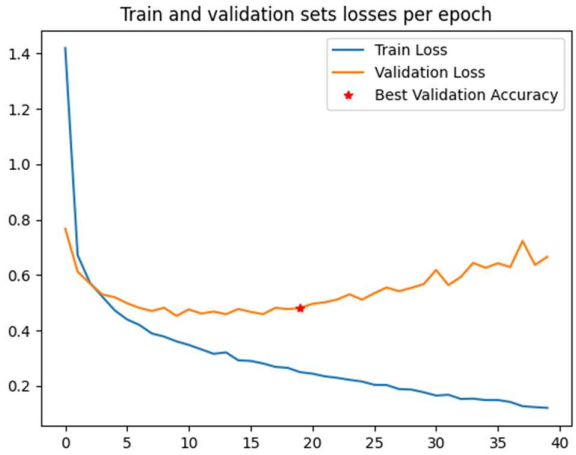
By the plot, we can clearly see there is overfitting in both cases and that's why the weight decay didn't help much. We use many epochs (250) on a very small dataset. Saying that, in the model w/o weight decay we got to 100% train accuracy. In other words, we got a full overfitting as opposed to the model with weight decay which prevented this from happening.

The plots also show that decaying the weight produced a better behavior during training.

We predict that using more data and less epochs will highly complement the weight decay model.

Function 9 - four_hidden_layers_adam_early_stopping (32 neurons):





Test Accuracy: 0.8319