COURSE NAME: Hillel - Machine Learning. Lesson 6. (16.03.2023 19:15,)Overfit. Regularization (Batch Normalization, Dropout)
LECTURER: Yevhen Krasnokutsky
STUDENT: Maryna Lysenko

HW 6 REPORT “Experiment with multiclass classification (MLP) for MNIST data set”.

- **Include dropout layers +**
- **Include batch normalization layers +**
- **Include more layers +**
- **Experiment with activation functions +**
- **Experiment presence/absence of dropout and batch normalization +**
- **Experiment with batch size +**
- **Plot loss = f(epochs) for each experiment +**

Experiment 0. Base model

NN architecture:

MnistMlp(

(wih): Linear(in_features=784, out_features=200, bias=True)

(who): Linear(in_features=200, out_features=10, bias=True)

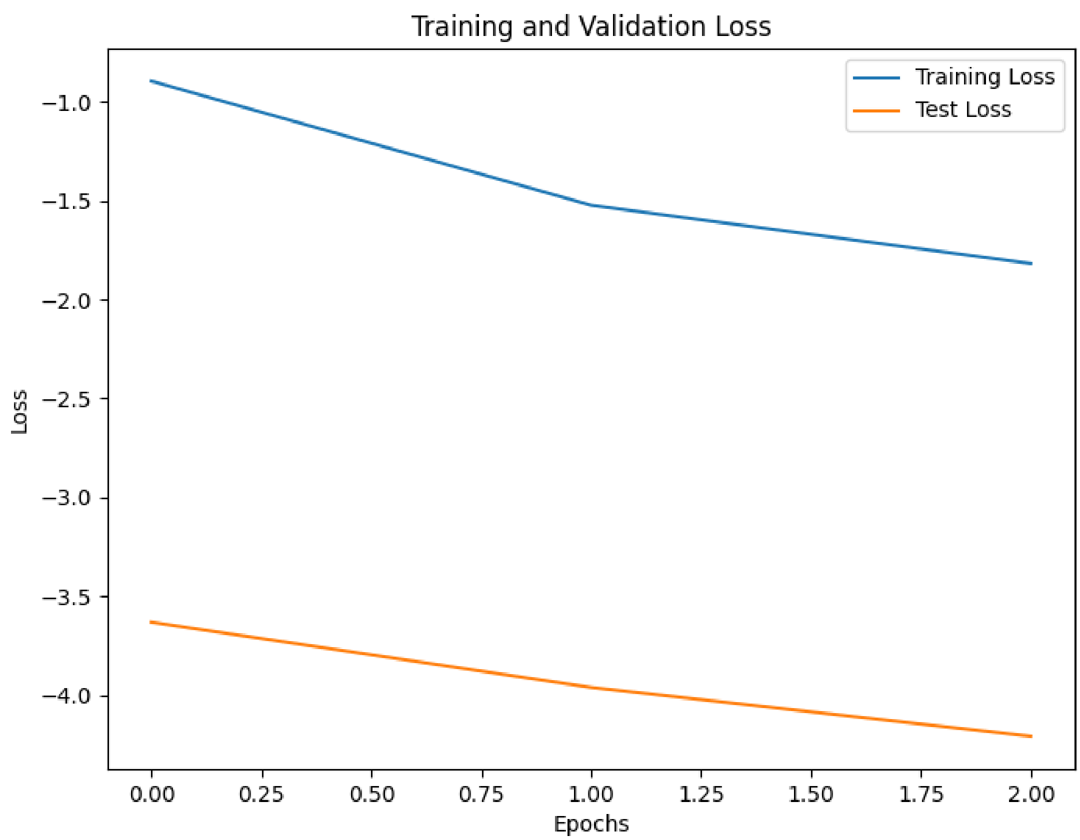
(activation): Sigmoid()

)

- learning_rate = 0.1
- batch_size = 10
- epochs = 3

Params: 159010

Test set: **Average loss**: 0.0149, **Accuracy**: 9557/10000 (96%)



Experiment 1. Include dropout layers

To add dropout to the model, we can simply insert torch.nn.Dropout() layers between the linear layers of the model. Dropout is a regularization technique used to prevent overfitting in neural networks by randomly dropping some of the neurons during training.

Params: 159430

MnistMlp(

(wih): Linear(in_features=784, out_features=200, bias=True)

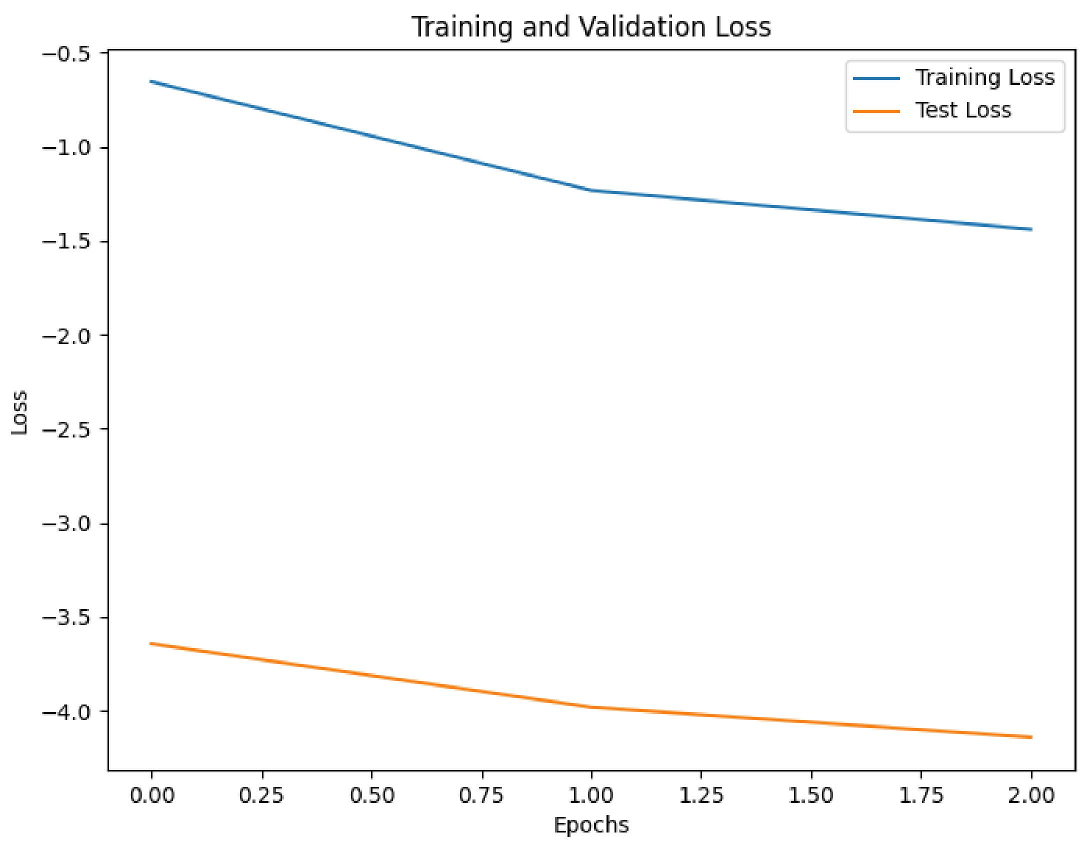
(dropout): Dropout(p=0.5, inplace=False)

(who): Linear(in_features=200, out_features=10, bias=True) (activation): Sigmoid()

)

- learning_rate = 0.1
- batch_size = 10
- epochs = 3

Test set: **Average loss**:0.0159, **Accuracy**: 9525/10000 (95%)



Experiment 2. Include batch normalization layers

MnistMlp(

(wih): Linear(in_features=784, out_features=200, bias=True)

(bn1): BatchNorm1d(200, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(dropout): Dropout(p=0.5, inplace=False)

(who): Linear(in_features=200, out_features=10, bias=True)

(bn2): BatchNorm1d(10, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

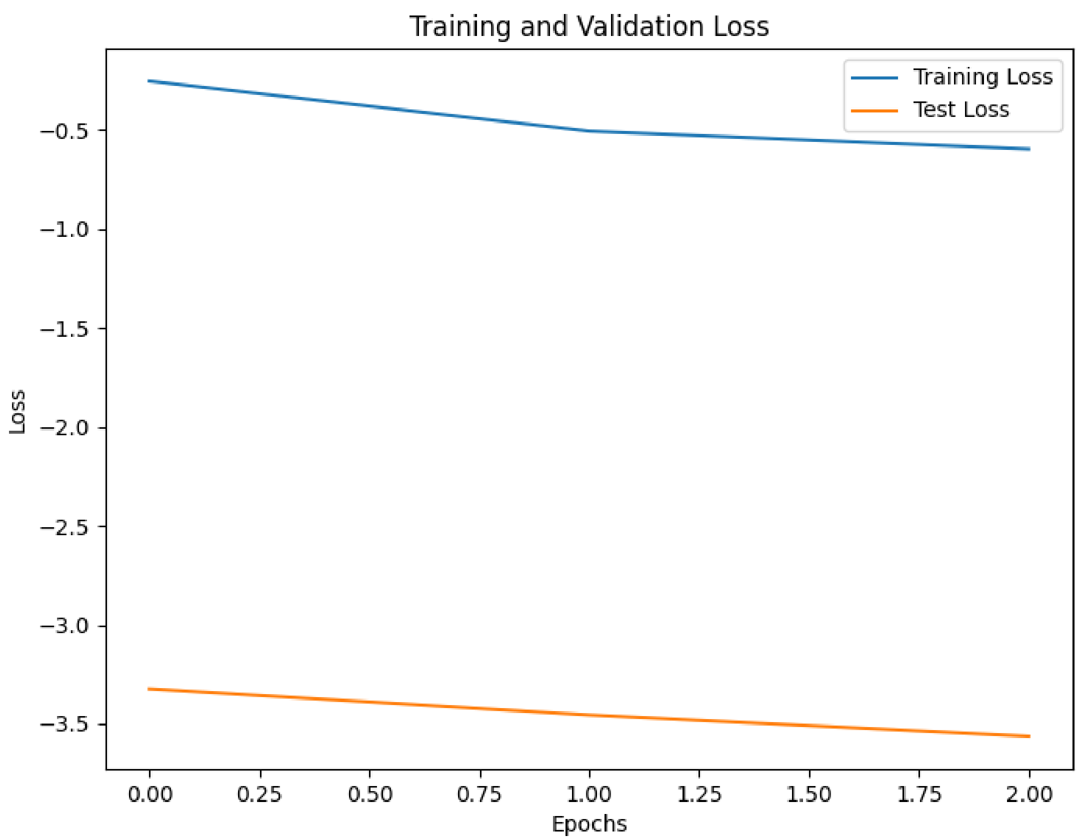
(activation): Sigmoid()

)

Params: 159430

- learning_rate = 0.1
- batch_size = 10
- epochs = 3

Test set: **Average loss:** 0.0283, **Accuracy**: 9152/10000 (92%)



Experiment 3. Include more layers

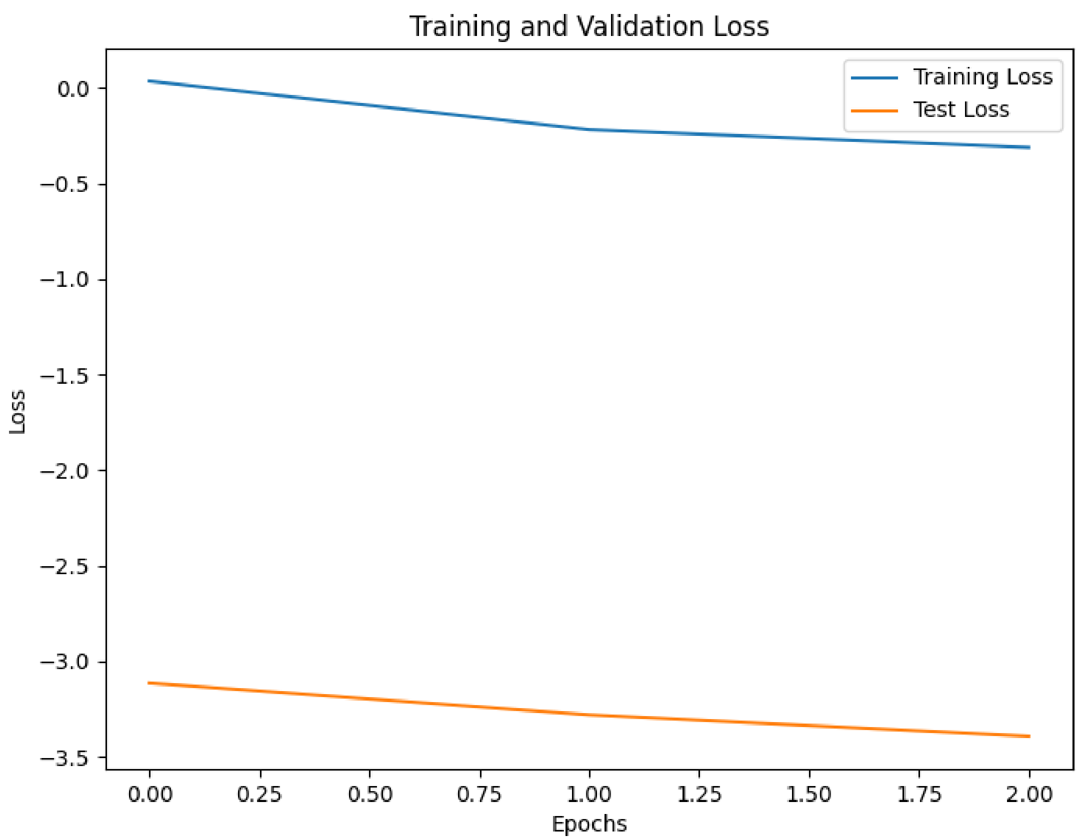
Params: 196030

MnistMlp(

(wih): Linear(in_features=784, out_features=200, bias=True)

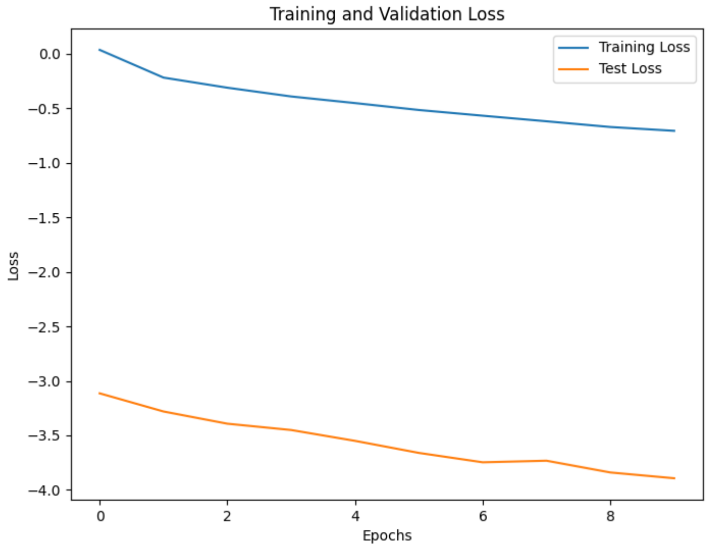
(bn1): BatchNorm1d(200, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

```
  (dropout1): Dropout(p=0.5, inplace=False)
  (hidden2): Linear(in_features=200, out_features=150, bias=True)
  (bn2): BatchNorm1d(150, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (dropout2): Dropout(p=0.3, inplace=False)
  (hidden3): Linear(in_features=150, out_features=50, bias=True)
  (bn3): BatchNorm1d(50, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (dropout3): Dropout(p=0.2, inplace=False)
  (who): Linear(in_features=50, out_features=10, bias=True)
  (bn4): BatchNorm1d(10, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
  (activation): Sigmoid()
)
Test set: Average loss: 0.0336, Accuracy: 9040/10000 (90%)
```



Experiment 3.1 More Epoch, more batch size, relu

Epoch = 10

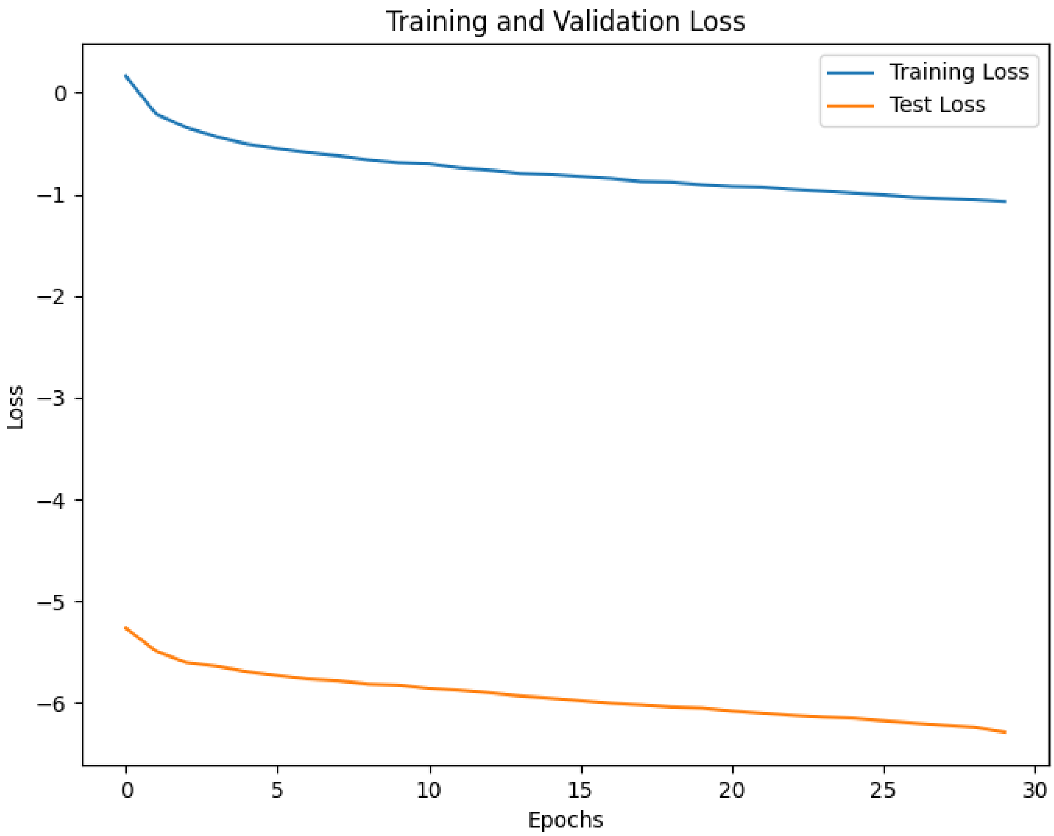


Test set: **Average loss:** 0.0182, **Accuracy**: 9483/10000 (95%)

Epoch = 30

Test set: **Average los**s: 0.0123, **Accuracy**: 9677/10000 (97%)

batch_size = 100



Test set: Average loss: 0.0019, Accuracy: 9448/10000 (94%)

Relu

MnistMlp(

(wih): Linear(in_features=784, out_features=200, bias=True)

(bn1): BatchNorm1d(200, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(dropout1): Dropout(p=0.5, inplace=False)

(hidden2): Linear(in_features=200, out_features=150, bias=True) (bn2): BatchNorm1d(150, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(dropout2): Dropout(p=0.3, inplace=False)

(hidden3): Linear(in_features=150, out_features=50, bias=True)

(bn3): BatchNorm1d(50, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(dropout3): Dropout(p=0.2, inplace=False)

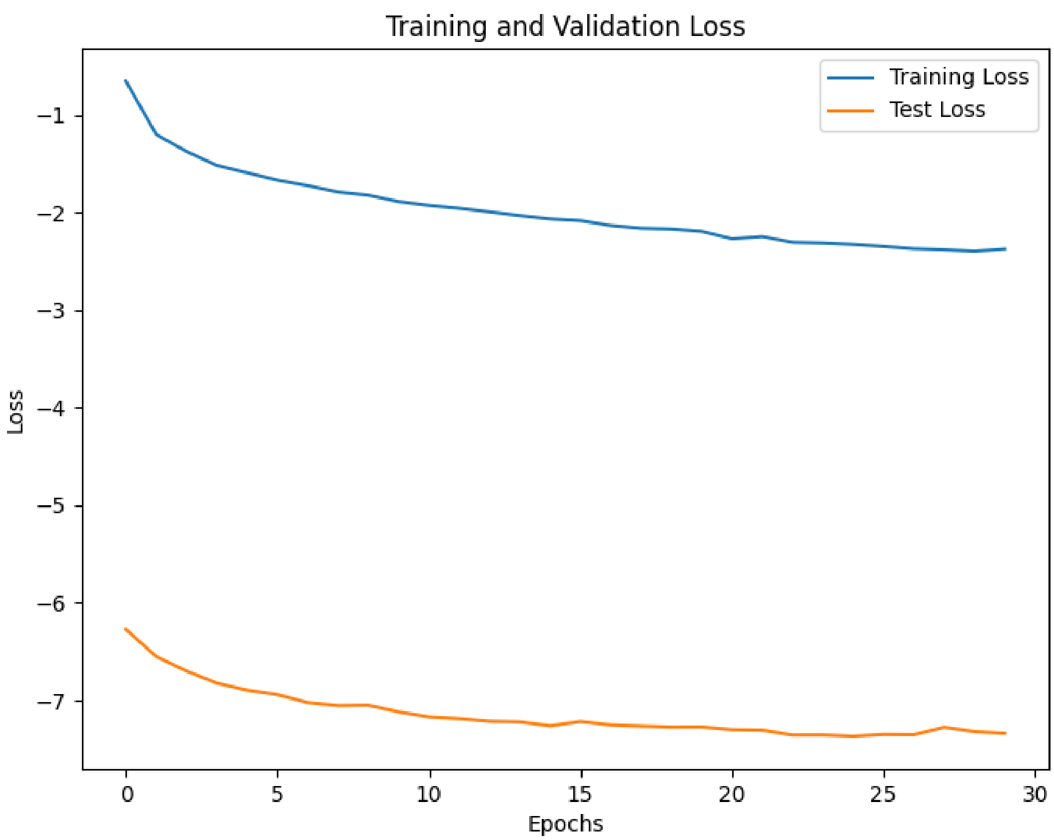
(who): Linear(in_features=50, out_features=10, bias=True)

(bn4): BatchNorm1d(10, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)

(activation): ReLU()

)

Test set: **Average loss**: 0.0007, **Accuracy**: 9821/10000 (98%)



Conclusions:

1. Dropout layers have a regularizing effect on the model and can help prevent overfitting. Experiment 1 showed that adding dropout layers decreased the accuracy slightly but may prevent overfitting in the model.
2. Batch normalization layers help to stabilize the training of the model and improve generalization performance. However, in Experiment 2, the model's accuracy decreased slightly after adding batch normalization layers.
3. Increasing the number of layers in the model does not necessarily lead to better performance. Experiment 3 showed that adding more layers decreased the model's accuracy, indicating that the model may have become too complex.

4. Experiment 3.1 showed that increasing the number of epochs can improve the model's accuracy, but increasing the batch size did not significantly impact the model's performance.
5. The choice of activation function can affect the model's performance. Experiment 3.1 using the ReLU activation function improved the accuracy of the model, indicating that ReLU may be a better choice than the sigmoid function used in the other experiments.