Deep Learning in Practice - TP1 Report
Hyperparameters and Training Basics with PyTorch

Jean-Vincent Martini - Adonis Jamal

January 23, 2026

1 Multi-Classification Problem

For this problem, we want to perform multi-class classification on the USPS dataset using a neural network implemented in PyTorch. This dataset consists of grayscale images of handwritten digits (0-9) with a resolution of 16x16 pixels.

Our neural network model consists of identical 2d convolutional layers with a kernel size of 3x3, followed by a non-linear activation function and max-pooling layers. The final layer is a fully connected layer that maps the extracted features to the 10 output classes corresponding to the digits 0-9. We chose this architecture because convolutional layers are well-suited for image data, as they can effectively capture spatial hierarchies and patterns in the images.

To tune the hyperparameters of the model, we will proceed by ablation studies, meaning that we will vary one hyperparameter at a time while keeping the others fixed to their default values. This approach allows us to isolate the effect of each hyperparameter on the model's performance and to gain time compared to a full grid search, as we will discuss later in section 3. However, it is important to note that this method may not capture interactions between hyperparameters, which could lead to suboptimal configurations. The hyperparameters we will explore include the number of layers, the number of neurons in each hidden layer, the activation function, the batch size, the learning rate, the number of epochs, the optimizer, and the loss function. The default configuration of the model is as follows: 2 hidden layers with 16 neurons each, ReLU activation function, batch size of 64, learning rate of 0.001, 30 epochs, Adam optimizer, and Mean Squared Error (MSE) loss function. The results of the ablation studies for each hyperparameter are presented in the Appendix, section A. For each configuration, we report the training accuracy, validation accuracy, and training time to provide a comprehensive view of the model's performance.

From the results, we decide to choose the final configuration of the model based on the best validation accuracy achieved for a reasonable training time. For the final configuration, we choose the same architecture as the default one, but with a learning rate of 0.01 and the loss function set to Cross-Entropy. This configuration achieves a training accuracy of 99.98%, a validation accuracy of 98.30%, and crucially a test accuracy of 95.81%. The test accuracy is obtained by evaluating the final model on the test set, which was not used during training or validation in section A. This metric therefore provides an unbiased estimate of the model's generalization performance on unseen data.

2 Regression Problem

In this problem, we aim to perform regression on a synthetic dataset using a neural network implemented in PyTorch. The task involves predicting the sale price of houses based on their overall quality, their date of construction, their total basement area, and their above-ground living area. The dataset consists of 1460 samples, each with those 4 features and the corresponding sale price as the target variable. With this dataset, it was absolutely necessary to perform feature normalization, as the features are on very different scales. This step ensures that the loss does not attain extremely high values, as shown in the code example provided.

Our neural network model consists of multiple indentical fully connected linear layers, each followed by a non-linear activation function. The final layer maps the extracted features to a single output value representing the predicted sale price. We proceed with ablation studies the same way as in section 1, considering the same hyperparameters. The other loss function we consider is the Gaussian log-likelihood loss, used when we instead predict both the mean and the variance of the target variable, assuming a Gaussian distribution. In this case, the final layer of the model therefore outputs two values per sample. The default configuration of the model is as follows: 3 hidden layers with 16 neurons each, ReLU activation function, batch size of 32, learning rate of 0.001, 100 epochs, Adam optimizer, and MSE loss function. As before, the results of the ablation studies for each hyperparameter are presented in the Appendix, section B.

Here, since the training time are relatively small, we choose the final configuration of the model based on the best validation loss achieved. For the final configuration, we decided to keep the same architecture as the default one, but with a batch size of 16. With this configuration, we achieve a training loss of 0.1348, a validation loss of 0.1476, and a test loss of 0.1851.

3 Observations and Trade-offs to Consider When Training Deep Learning Models

A Problem 1: Results for Different Hyperparameters

The symbol * indicates the value used in the default configuration.

Impact of the number of layers:

Number of Layers	Train Accuracy (%)	Validation Accuracy (%)	Train Time (s)
1	98.57	96.82	7.91
2*	99.08	97.44	14.52
3	99.22	97.37	18.26
5	97.83	96.90	24.69

Impact of the number of neurons in hidden layers:

Hidden Layer Size	Train Accuracy (%)	Validation Accuracy (%)	Train Time (s)
8	97.72	96.13	11.62
16*	99.08	97.44	14.52
32	99.68	98.14	18.94
64	99.90	98.92	29.29

Impact of the activation function:

Activation Function	Train Accuracy (%)	Validation Accuracy (%)	Train Time (s)
ReLU*	99.08	97.44	14.52
Sigmoid	96.83	94.58	14.28
Tanh	99.72	97.60	14.79
Leaky ReLU	98.77	97.37	14.39

Impact of the batch size:

Batch Size	Train Accuracy (%)	Validation Accuracy (%)	Train Time (s)
16	99.62	98.76	18.74
32	99.05	96.82	15.78
64*	99.08	97.44	14.52
128	98.18	96.59	13.57

Impact of the learning rate:

Learning Rate	Train Accuracy (%)	Validation Accuracy (%)	Train Time (s)
0.001*	99.08	97.44	14.52
0.01	99.43	98.14	14.36
0.1	16.23	17.04	14.21
1	16.23	17.04	14.34

Impact of the number of epochs:

Number of Epochs	Train Accuracy (%)	Validation Accuracy (%)	Train Time (s)
10	97.27	95.82	4.87
30*	99.08	97.44	14.52
50	99.50	97.68	24.05
100	99.50	97.13	47.41

Impact of the optimizer:

Optimizer	Train Accuracy (%)	Validation Accuracy (%)	Train Time (s)
Adam*	99.08	97.44	14.52
RMSprop	99.17	97.29	14.27

Impact of the loss function:

Loss Function	Train Accuracy (%)	Validation Accuracy (%)	Train Time (s)
MSE*	99.08	97.44	14.52
Cross-Entropy	99.05	97.68	14.35

B Problem 2: Results for Different Hyperparameters

The symbol * indicates the value used in the default configuration. The reported losses are Mean Squared Error (MSE) losses, even if the loss function used during training was different.

Impact of the number of layers:

Number of Layers	Train Loss	Validation Loss	Train Time (s)
1	0.1566	0.1597	0.56
2	0.1520	0.1616	0.81
3*	0.1447	0.1469	0.91
5	0.1414	0.1540	1.11

Impact of the number of neurons in hidden layers:

Hidden Layer Size	Train Loss	Validation Loss	Train Time (s)
8	0.1533	0.1693	0.76
16*	0.1447	0.1469	0.91
32	0.1477	0.1456	0.89
64	0.1173	0.1370	1.08
128	0.1098	0.1347	1.32

Impact of the activation function:

Activation Function	Train Loss	Validation Loss	Train Time (s)
ReLU*	0.1447	0.1469	0.91
Sigmoid	0.1583	0.2060	0.84
Tanh	0.145	0.1875	0.85
Leaky ReLU	0.1467	0.1773	0.84

Impact of the batch size:

Batch Size	Train Loss	Validation Loss	Train Time (s)
8	0.1311	0.1525	2.86
16	0.1232	0.1386	1.53
32*	0.1447	0.1469	0.91
64	0.1564	0.1823	0.51

Impact of the learning rate:

Learning Rate	Train Loss	Validation Loss	Train Time (s)
0.0001	0.1666	0.2005	0.82
0.001*	0.1447	0.1469	0.91
0.01	0.1358	0.1730	0.84
0.1	0.1640	0.2127	0.82

Impact of the number of epochs:

Number of Epochs	Train Loss	Validation Loss	Train Time (s)
50	0.1498	0.1594	0.43
100*	0.1447	0.1469	0.91
200	0.1270	0.1532	1.72
500	0.0970	0.1346	4.22
1000	0.0850	0.1389	8.43

Impact of the optimizer:

Optimizer	Train Loss	Validation Loss	Train Time (s)
Adam*	0.1447	0.1469	0.91
RMSprop	0.1443	0.1677	0.87

Impact of the loss function:

Loss Function	Train Loss	Validation Loss	Train Time (s)
MSE*	0.1447	0.1469	0.91
Gaussian log-likelihood	_	0.1589	0.97