MARIA KOILALOU *JUSTINE LE BOURG*

Artificial Intelligence

Laboratory 5

Variant 1

This report presents an analysis and an evaluation of the code developed for the training of a classification model of image by using the MNIST data set. The objective of this project was to explore different combinations of hyperparameters and loss functions in order to find the optimal configuration for the model. The report will highlight the structure of the code, its clarity, its performance and the results obtained.

Data preparation is the first step in the code. The necessary libraries are imported, including Numpy, Matplotlib and Tensorflow. Then, the MNIST data set is imported using the Keras.datasets.mnist function, and the data are divided into training and testing sets.

The data is pre-treated to adapt it to the input of the MLP model. The images are redesigned in vectors and the values of the pixels are standardized between 0 and 1. In addition, class labels are converted into categories using the to_categorical function of Keras.utils. Then, the training sets are divided into training and validation sets using the train_test_Split function of Sklearn.model_selection.

The architecture of the MLP model is then defined. The model is created using Keras.models.Sequential and includes an input layer with 784 knots (corresponding to the size of the MNIST images after remodeling), one or two layers hidden with a variable number of nodes (defined by hyperparameters), and an output layer with 10 nodes corresponding to the classes of the digit from 0 to 9.

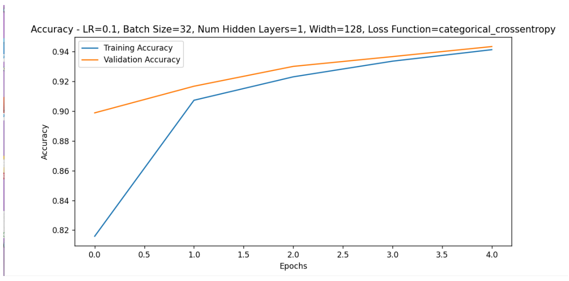
The main part of the code is to explore hyperparameters. The different combinations of hyperparameters, such as the learning rate, the size of the lot, the number of hidden layers, the width of the hidden layers and the loss functions, are defined in lists.

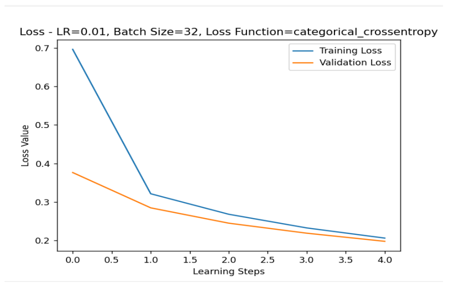
A nested loop is used to train and assess the model for each combination of hyperparameters. With each iteration, the model is rebuilt with the corresponding hyperparameters, compiled with the loss function Categorical_crossentropy and the SGD optimizer, then adjusted to training data. The final validation precision is recorded with the results.

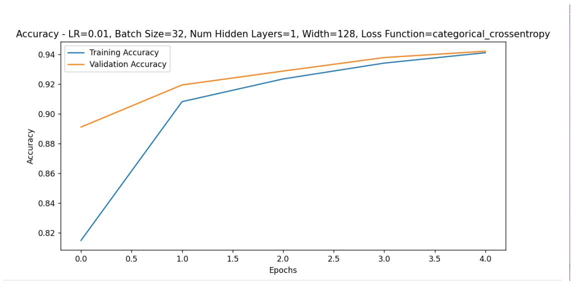
Finally, the results are displayed. Graphics of loss and accuracy of training and validation are traced for each combination of hyperparameters. The best result is identified according to the maximum validation accuracy and is displayed at the end of the report.

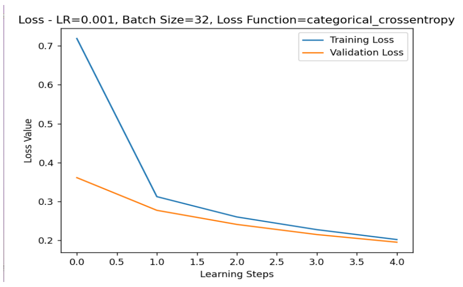
OUTPUT

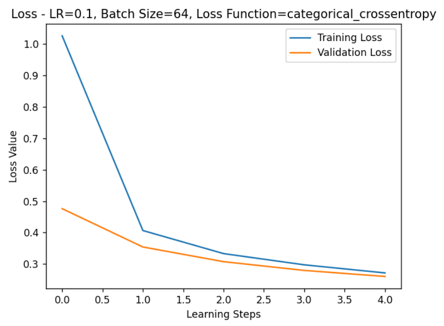
The output provide graphs of training and validation accuracy as training and validation loss

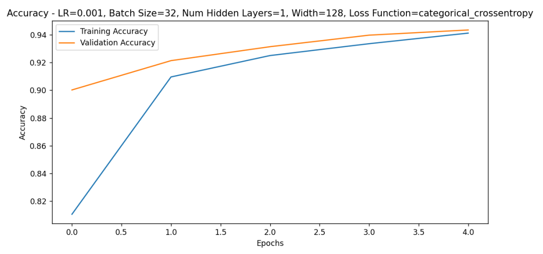


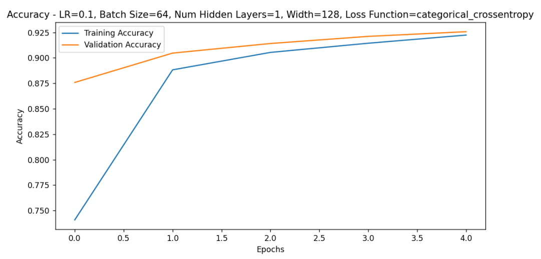


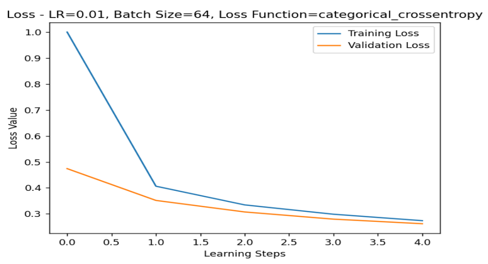


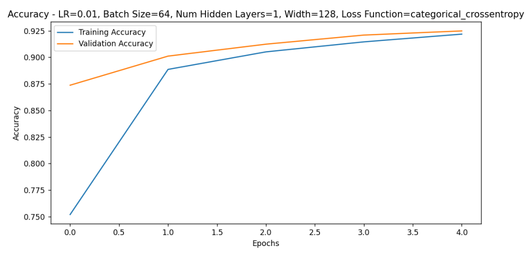


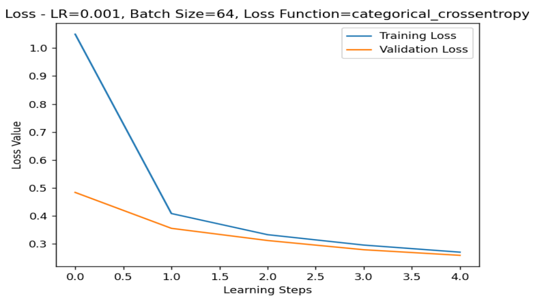


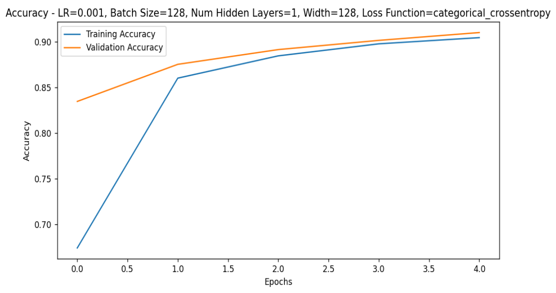


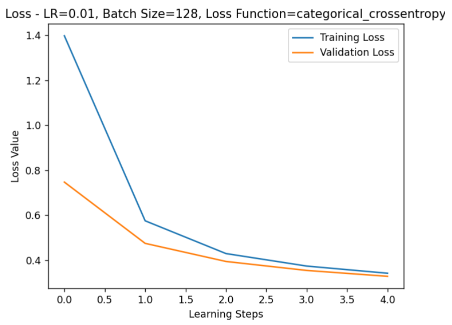


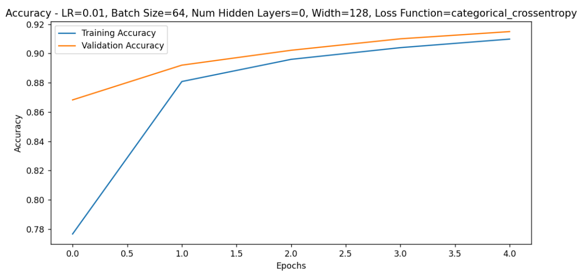


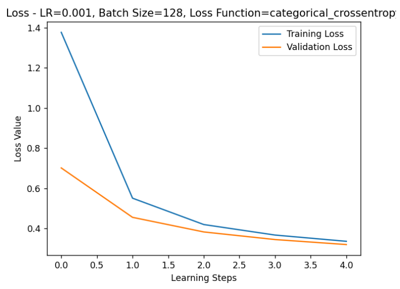

Accuracy - LR=0.01, Batch Size=64, Num Hidden Layers=1, Width=128, Loss Function=categorical_crossentropy
Training Accuracy
Validation Accuracy
Accuracy
Epochs

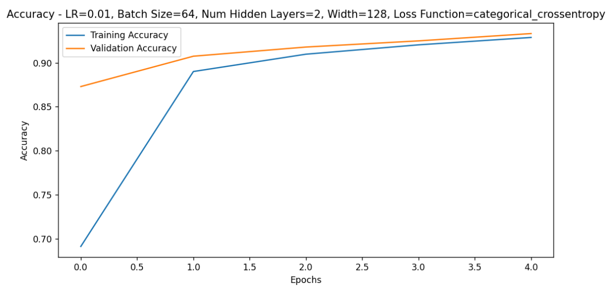

Loss - LR=0.001, Batch Size=64, Loss Function=categorical_crossentropy
Training Loss
Validation Loss
Loss Value
Learning Steps

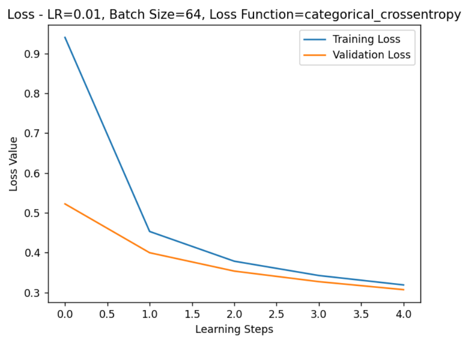

Accuracy - LR=0.001, Batch Size=128, Num Hidden Layers=1, Width=128, Loss Function=categorical_crossentropy
Training Accuracy
Validation Accuracy
Accuracy
Epochs

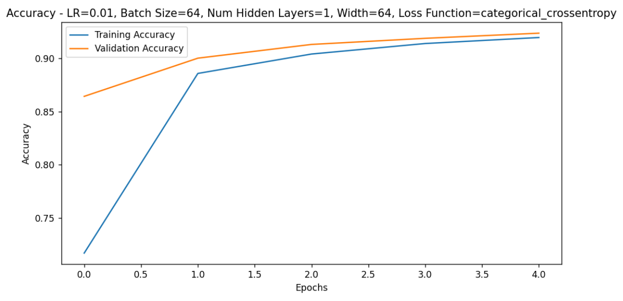

Loss - LR=0.01, Batch Size=128, Loss Function=categorical_crossentropy
Training Loss
Validation Loss
Loss Value
Learning Steps

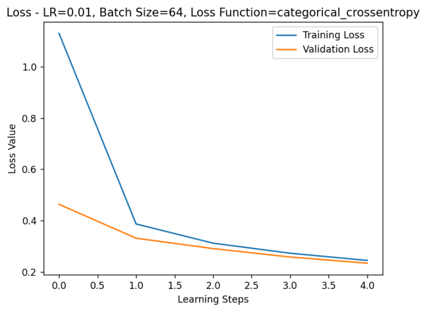

Accuracy - LR=0.01, Batch Size=64, Num Hidden Layers=0, Width=128, Loss Function=categorical_crossentropy
Training Accuracy
Validation Accuracy
Accuracy
Epochs

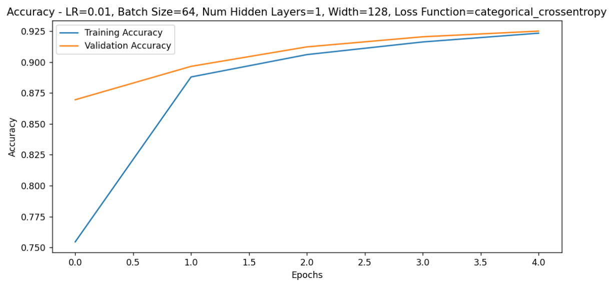

Loss - LR=0.001, Batch Size=128, Loss Function=categorical_crossentrop
Training Loss
Validation Loss
Loss Value
Learning Steps

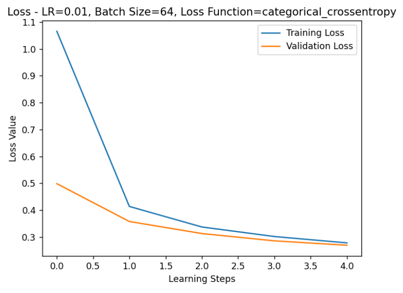

Accuracy - LR=0.01, Batch Size=64, Num Hidden Layers=2, Width=128, Loss Function=categorical_crossentropy
Training Accuracy
Validation Accuracy
Accuracy
Epochs

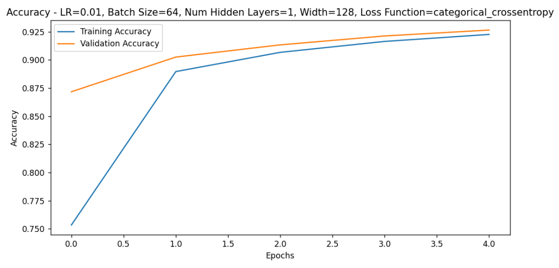

Loss - LR=0.01, Batch Size=64, Loss Function=categorical_crossentropy
Training Loss
Validation Loss
Loss Value
Learning Steps

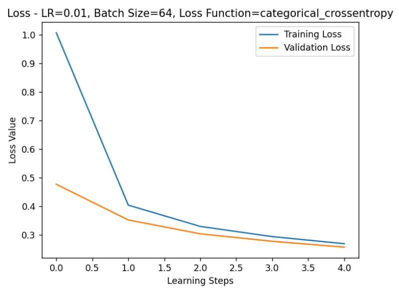

Accuracy - LR=0.01, Batch Size=64, Num Hidden Layers=1, Width=64, Loss Function=categorical_crossentropy
Training Accuracy
Validation Accuracy
Accuracy
Epochs

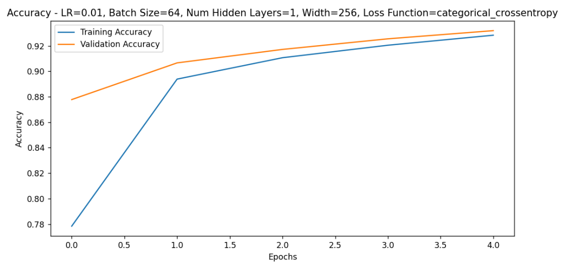

Loss - LR=0.01, Batch Size=64, Loss Function=categorical_crossentropy
Training Loss
Validation Loss
Loss Value
Learning Steps

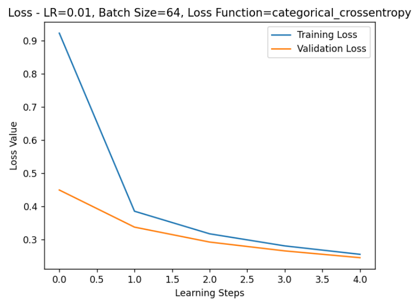

Accuracy - LR=0.01, Batch Size=64, Num Hidden Layers=1, Width=128, Loss Function=categorical_crossentropy
Training Accuracy
Validation Accuracy
Accuracy
Epochs

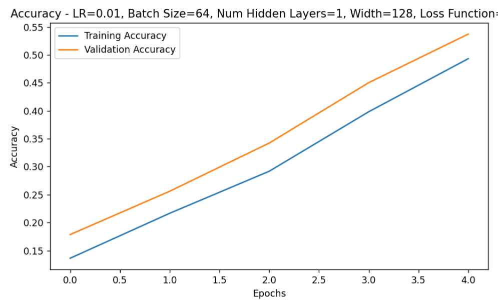

Loss - LR=0.01, Batch Size=64, Loss Function=categorical_crossentropy
Training Loss
Validation Loss
Loss Value
Learning Steps

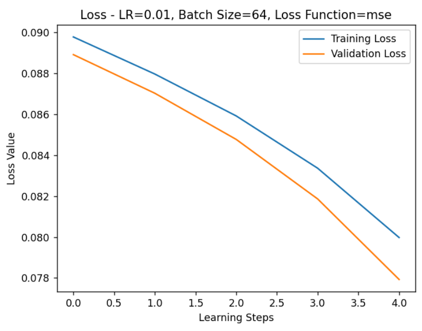

Accuracy - LR=0.01, Batch Size=64, Num Hidden Layers=1, Width=128, Loss Function=categorical_crossentropy
Training Accuracy
Validation Accuracy
Accuracy
Epochs

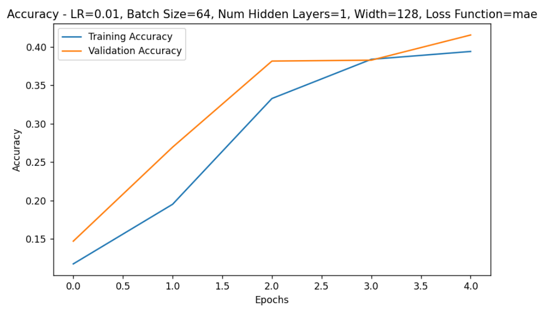

Loss - LR=0.01, Batch Size=64, Loss Function=categorical_crossentropy
Training Loss
Validation Loss
Loss Value
Learning Steps

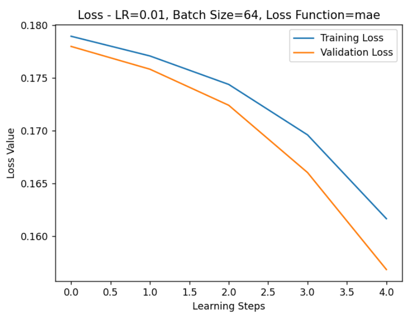

Accuracy - LR=0.01, Batch Size=64, Num Hidden Layers=1, Width=256, Loss Function=categorical_crossentropy
Training Accuracy
Validation Accuracy
Accuracy
Epochs

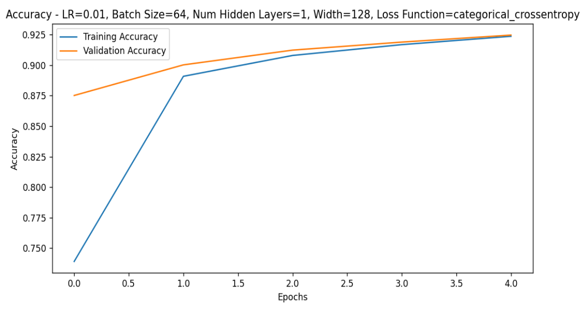

Loss - LR=0.01, Batch Size=64, Loss Function=categorical_crossentropy
Training Loss
Validation Loss
Loss Value
Learning Steps

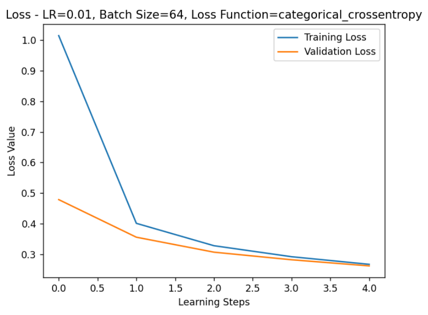


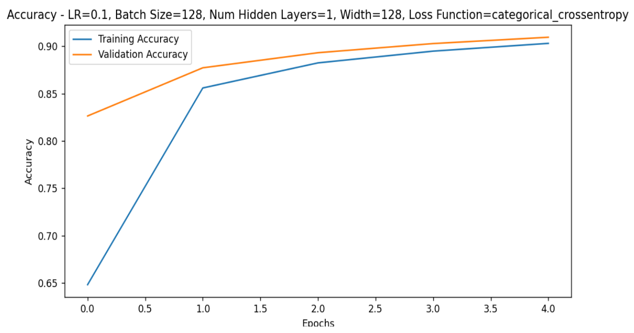


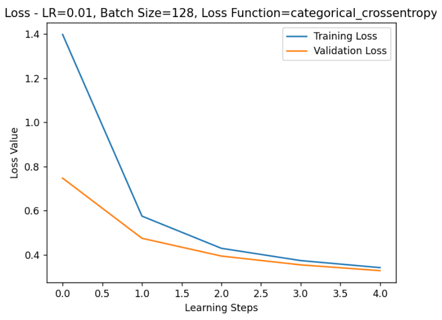












After analyzing the results of our different training examples, here are the conclusion we can draw:

The use of a Learning rate of 0.1 and a 32 batch size gave good results in terms of loss and accuracy on the validation set. However, this took more time . On the other hand, when we used a Learning rate of 0.01 and a 64 batch size, we obtained similar results, but with a shorter training time.
It is also important to note that we used a single level of depth and a hidden layer width of 128 for most of the models because testing all the parameters for all the different values was very time consuming and the code could not finish in a specific time period..

Regarding the loss function, the function "Categorical_crossentropy" has been mainly used, which is commonly used for multi-class classification problems, and it has given good results in our case.
We checked each 3 values of the hyperparameters with the others being fixed except for learning rate and batch size in which we checked all possible combinations.

In conclusion,the models with a Learning rate of 0.1 and a batch size of 32 gave good results in terms of loss and accuracy on the validation set. However, the models with a learning rate of 0.01 and a 64 batch size also gave similar results, but with shorter training times in time. So we can choose one of these models based on the training and resource time constraints.