COMP 551 - Mini Project 3

Nicolas Fertout, Luna Dana, Anthony Kumar

March 2022

Abstract

This project involved implementing a multilayer perceptron model from scratch to make predictions on data from the Fashion-MNIST dataset, and analyzing the effect of changing network depth and activation functions on the accuracy of the model, and comparing the performance of the MLP model with the CNN model to classify image data. We found increasing depth increased our model's accuracy, and the CNN model performed significantly better than the MLP model.

1 Introduction

In this report, we investigate and analyze the performance between of different variation of multilayers perceptron and a Convutional Neural Network on the fashion-MNIST dataset. (described in the Datasets section). We used an MLP with different layers and activation functions, which were implemented in the Python programming language, for classifying new instances of data into one of their 10 respective classes. The CNN model we used was PyTorch's implementation. Our focus was on identifying the most important parameters for the models, as well as the model that yields the highest accuracy.

Overall, we found that the CNN model gives a better fit compared to the MLP model on classifying images (using 2 hidden layer especially). We also discovered that, for both the CNN and MLP models, increasing the network depth from 0 to 1 to 2 increases the model's accuracy on the test set.

The Zalando research center has published multiple results obtained by the MLP classifier. The best of them is a 87.4% accuracy obtained by a ReLu activation function and 2 hidden layers with respective sizes of 100 and 10. Another Implementation of the MLP model gives better results. Indeed, they had a 88.4% accuracy with 1 input layer, 3 hidden layers and 1 output layer (fully connected neural network with architecture 728-256-128-100-10).

We will start by explaining what our data consists of and how it was cleaned appropriately in order to be used in our models, then we will give some insights on the methods and design choices made during our project, to finish with a presentation of our results and a discussions of our findings.

2 Datasets

2.1 Data Description

The fashion MNIST dataset consists of 70000 instances (60000 train and 10000 test) which are black and white images of size 28x28. They represent picture of different type of clothing pieces such as trousers, dress, shoes etc. This data set is provided by the Zalando research branch. Each pixel has a single value associated, indicating the lightness or darkness of it, with higher numbers meaning darker pixels. This pixel-value is an integer going from 0 to 255.

2.2 Data Cleaning

Our first task is to normalize the data in order for each picture to have the same taints of grey. That is, if one picture is very dark and another is very bright, we make their pixel to be on the same scale.

For each pixel, we subtract the mean of the same pixel in all other images of the dataset and divide it by the standard deviation.

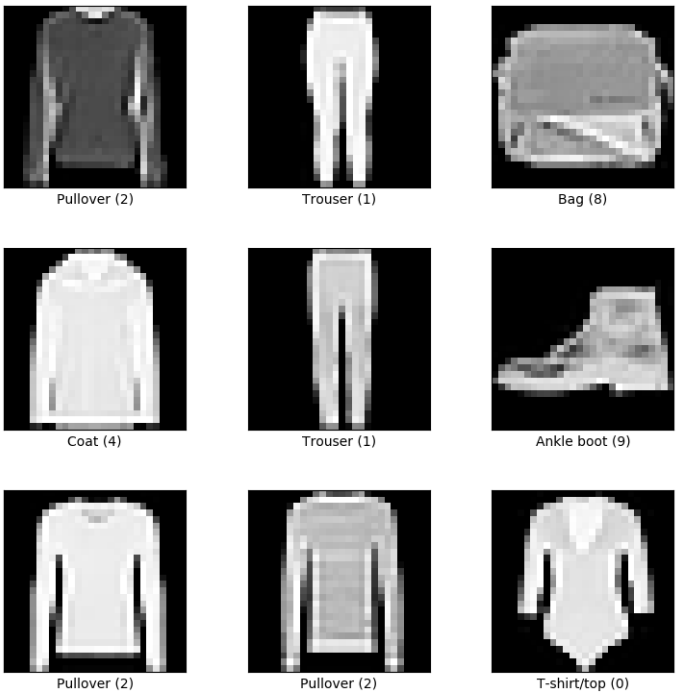

Figure 1: Sample of the fashion-MNIST dataset

2.3 Data Analysis

The data has 10 different categories which are labelled from 0 to 9 which are the following : 0 : T-shirt/top, 1 : Trouser, 2 : Pullover, 3 : Dress, 4 : Coat, 5 : Sandal, 6 : Shirt, 7 : Sneaker, 8 : Bag, 9 : Ankle boot.

There are exactly 6000 instance of each type. Some concerns with the data may occur concerning some piece of clothing. Indeed, dresses have different shapes and variation of colours which could lead to bad classification. Moreover, some pants pictures are taken from the side and hence this class also have a large spectrum of pictures that could raise errors. The data shares the same image size and structure of training and testing splits as the MNIST original library.

3 Design Choices

- Depth of model: The more depths tho model has, the more accurate it is. However, the marginal accuracy of adding more hidden layers is maybe not worth the time and space complexity it adds. A good trade-off needs to be found to avoid too simple or too complex models.

- Activation function: We used different activation functions in our models (Leaky ReLu, ReLu and Tanh) each giving different accuracies. For a multiclass classification problem with image data, Leaky-ReLU seemed to be the most adequate choice. We will discuss the details in result

- Data Augmentation: In order to have more data, we have implemented some data augmentation technique with the help of this website which gave good techniques and snippets of code. The main one was to flip horizontally every image to have double the size of our actual dataset. The image augmentation introduced to the training data does improve the accuracy for complex-enough models, although it takes more epochs for the model to improve itself.

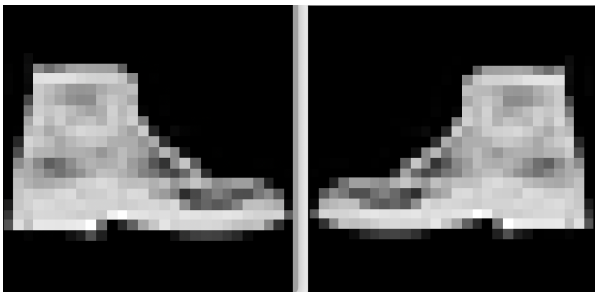

Figure 2: Example of a horizontally flipped image

- Dropout: This regularization methods consists in randomly removing a subset of units during the training phase which is one of the most used and performing technique in MLP.

4 Results

We ran several experiments, on MLP and CNN, using different layers, activation functions, normalized/unnormalized data and augmented data. Our results are reported below.

# of Hidden Layers	ReLu Accuracy	Leaky-ReLU Accuracy	Tanh Accuracy
0	77.54%	79.82%	77.57%
1	82.70%	83.91%	82.48%
2	82.83%	84.10%	82.50%

Table 1: Average accuracy on the MLP classifier using a varying number of 128-unit hidden layers across the ReLu, Tanh, and Leaky ReLu activation functions.

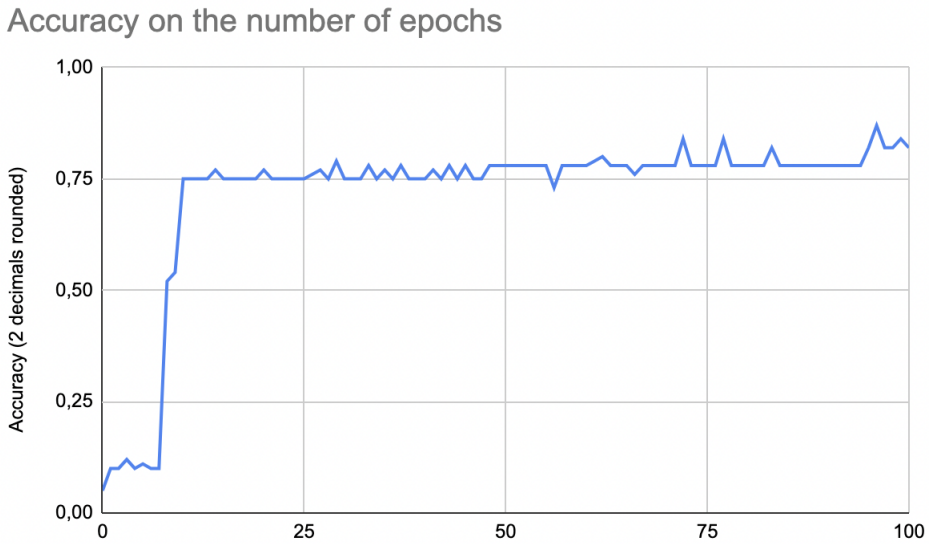

Figure 3: Accuracy on the number of epoch with a ReLu activation function

# of Hidden Layers	normalized images	unnormalized images
0	77.54%	76.69%
1	82.70%	79.17%
2	82.83%	79.34%

Table 2: Average accuracy on the MLP classifier using a varying number of 128-unit hidden layers on normalized and unnormalized data.

# of Hidden Layers	Multilayer Perceptron	Convolutional Neural Network
0	77.54%	91.74%
1	82.70%	96.99%
2	82.83%	99.03%

Table 3: Average accuracy of the MLP & CNN classifiers using ReLU activations and a varying number of 128-unit hidden layers on the Fashion-MNIST dataset

# of Hidden Layers	Multilayer Perceptron with data augmentation
0	71.22%
1	81.96%
2	83.45%

Table 4: Average accuracy of the MLP & CNN classifiers using ReLU activations and a varying number of 128-unit hidden layers on the Fashion-MNIST dataset with double the dataset size (using data augmentation)

5 Discussions and Conclusion

When varying the number of 128-unit hidden layers in the MLP classifier from 0 hidden layers to 2 hidden layers all using ReLU activations, we get a significant performance increase going from 0 hidden layers to 1 hidden layer (approx. 5% increase in accuracy). However, going from 1 hidden layer to 2 hidden layers has a small performance increase. We can conclude that as the network gets deeper from 0 to 2 hidden layers, the performance of the model increases. This is expected because increasing the network's depth allows the model to be more expressive and capture more details from the data. However, eventually, increasing the depth too much may lead to overfitting of the data and decrease performance.

In comparing the MLP model's accuracy using three different activation functions (ReLU, Tanh, Leaky ReLU) throughout the network, we find that the ReLu MLP model has a very similar performance to the Tanh MLP model. The Leaky-ReLU MLP model seems to perform slightly better than both the ReLU and Tanh MLP models. This is expected since the Leaky-ReLU MLP model prevents neurons from dying and having 0 contribution which happens in the ReLU model since no weight actually hits 0. This allows for all parameters to have an effect on the output.

In training the MLP model with unnormalized images, using a varying number of 128-unit layer ReLU MLP model, we got a slight decrease in performance for 0,1 and 2 hidden layers as expected since this process normalizes the data to all be at the same scale. We also expected this rather small decrease in performance because seeing how our data is image data and that pixel values can only range from 0 to 255, we already have our data on a similar scale, so we do not expect a drastic change in performance when avoiding normalization for this data. Besides, we tried using data augmentation techniques during the training phase, which led to lower results for 0 and 1 layers and slightly better for 2 layers. This could be explained by the fact that, adding new vertically flipped data may be harder to grasp for less complex models, which in this case might have led to a higher bias for 0 and 1 layer.

Moreover, we observe a similar trend as for MLP in doing this experiment with the CNN model. When running the same experiments using a Convolutional Neural Network, (2 hidden-128 unit-layers & ReLU activations), the model achieved a near perfect accuracy on classifying the images. The CNN model performed significantly better than the MLP model in general for 0, 1 and 2 hidden layers. This is expected because the convolutional layers are able to capture data in 2 dimensions unlike MLPs, and knowing that our data is image data (2D data), we expect CNNs to perform better than MLPs for classifying image data.

To go further, we could have tried a larger number of layers and different units per layer, but the computation time was too long to run all of those experiments. We could also explore different settings for the CNN model which seems to be the more accurate (unsurprisingly), such as adding different values of padding, or different units and layers.

6 Group Member Contributions

- Everyone: Global planning of the project, discussion of design choices, writing of the report
- Luna Dana: Data cleaning script in Python, data augmentation and statistics, running experiments.
- Nicolas Fertout: Computation of gradient for 2 hidden layers, implementation of MLP (0, 2 hidden layers), running experiments.
- Anthony Kumar: Implementation of MLP (1 hidden layer) & CNN (PyTorch) models, running experiments