Student Name: Bérénice Le Glouanec

DVA453- Machine Learning with Big Data

Answers for Project ÖVN1 (Module 4)

Abstract

The aim of this project is to predict what piece of clothing is it among pictures. The system should be able to predict what is the prominent piece of clothing in a set of predefined classes of clothing. Using the Fashion-MNIST dataset, containing thousands of gray images from Zalando's articles. In order to solve this machine learning problem, we implement a soft Convolutional Neural Network based on LeNet-5 architecture model. Our CNN model consists of 5 convolutional layers, where each of them computes max pooling, batch normalization and drop out functions on the data, and 3 fully connected layers. During the training process, we perform backward propagation on the cross entropy loss, then we evaluate our model on the testing set based on their losses and their accuracy. Our results show that our trained model performs very well having an accuracy of 95.08% on the training set and 92.37% when tested.

Introduction

The main motivation behind this project is to have a better understanding of how it is possible to use recognition in images in Machine Learning. Various work has already been done is this domain, but my objective is purely educational and not for research purposes.

Using the Fashion-MNIST dataset, our goal is to make a prediction on a given image, and determine what piece of clothing is it among a set of pre-defined clothing class. Machine Learning is appropriate to solve this problem because from what the system will learn, it will be able to predict which kind of article (bag, trouser, coat …) is an object on a picture. As we can fine-tune the parameters of our model, it is possible to produce very accurate predictions. The Fashion-MNIST dataset represents 0.131596 GigaBytes, therefore it cannot be considered as Big Data. However, it is a reasonable first approach to conduct experiments in this domain.

This project is written in Python in a Jupyter Notebook and is executable on recent Kernels. The ZIP file contains the jupyter notebook file, the Fashion-MNIST dataset, and the model as "model_cnn.pt" file. All the necessary packages are imported in the notebook and therefore should not require further intervention by the user. In this project, we mainly use the following libraries: numpy, pytorch to create the model, sklearn for evaluating the model and matplotlib to visualize the output.

In the remainder of this report, we will discuss the work previously carried out on this subject with this dataset; we will detail how this project has been implemented, which structure of CNN was used; we will evaluate and discuss the results; and we will conclude on the achievements and learning outcomes of this project.

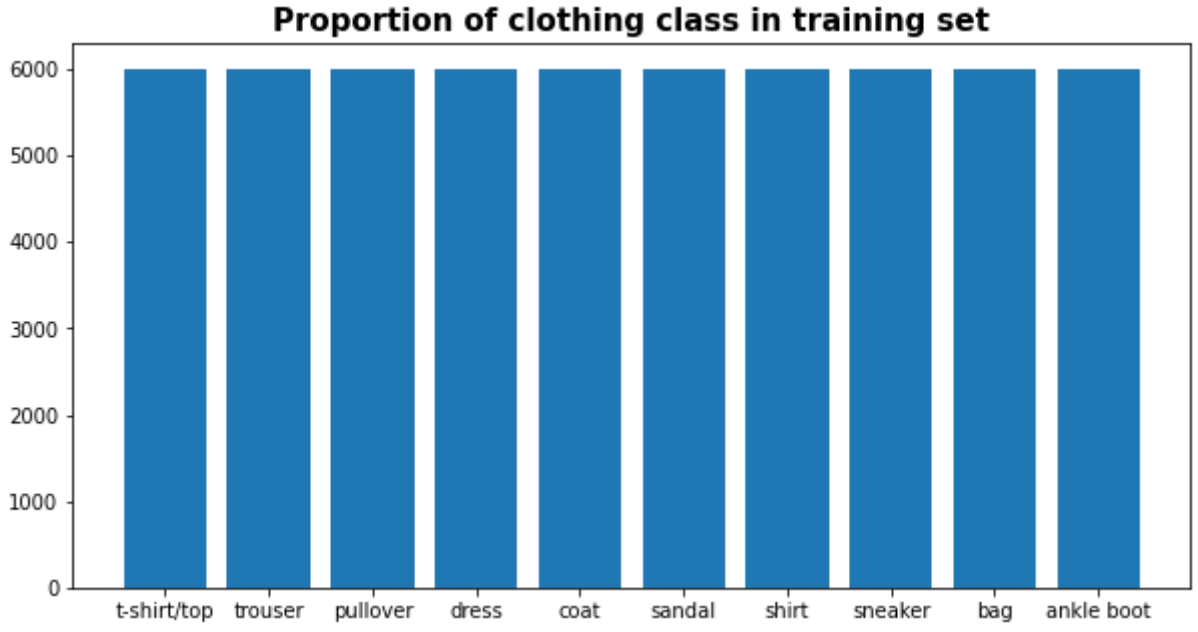
Implementation

The Fashion-MNIST dataset has been released in August 2017. It has been created in order to replace the original MNIST dataset, considered as too easy, overused and unable to represent modern machine learning tasks by the Machine Learning community. It consists of a training set of 60 000 images (and a testing set of 10 000 images), each of them containing 784 pixels (28 * 28 gray scale image) and 1 label. All

images are linked to a label that is represented as an integer from 0 to 9. There are 10 classes of clothing: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot. The images were cut so that the object is centered, on a black background, and there is no other information than the piece of clothing itself on the picture.

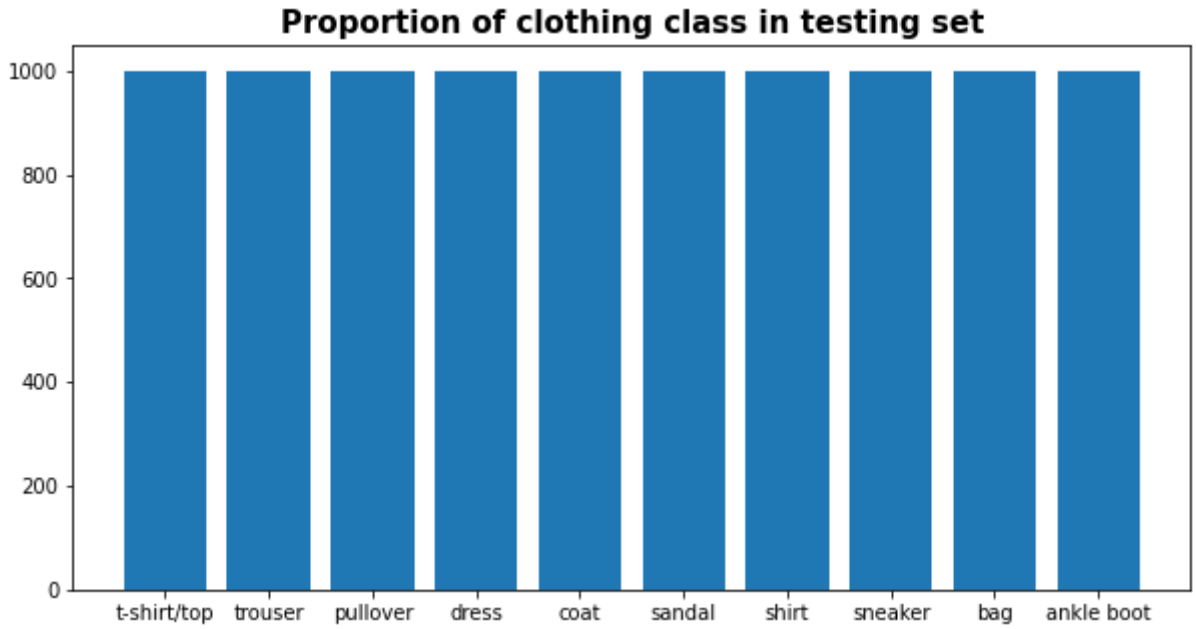
In order to exploit the Fashion-MNIST dataset, we convert train images, train labels, and test images, test labels “ubyte” files into train and test CSV files thanks to a script[1] provided by Ali Farhadi. Each line of a CSV file consists of a label (integer value from 0 to 9) and a list of 784 pixels (as integers). When combined all together in a 28 x 28 shape, they form a gray scale image of a garment.

Before processing any data, we checked the distribution of each label among the training and testing sets. A greater representation of one label compared to another one would lead to a better accuracy for the first label and worse for the second one.



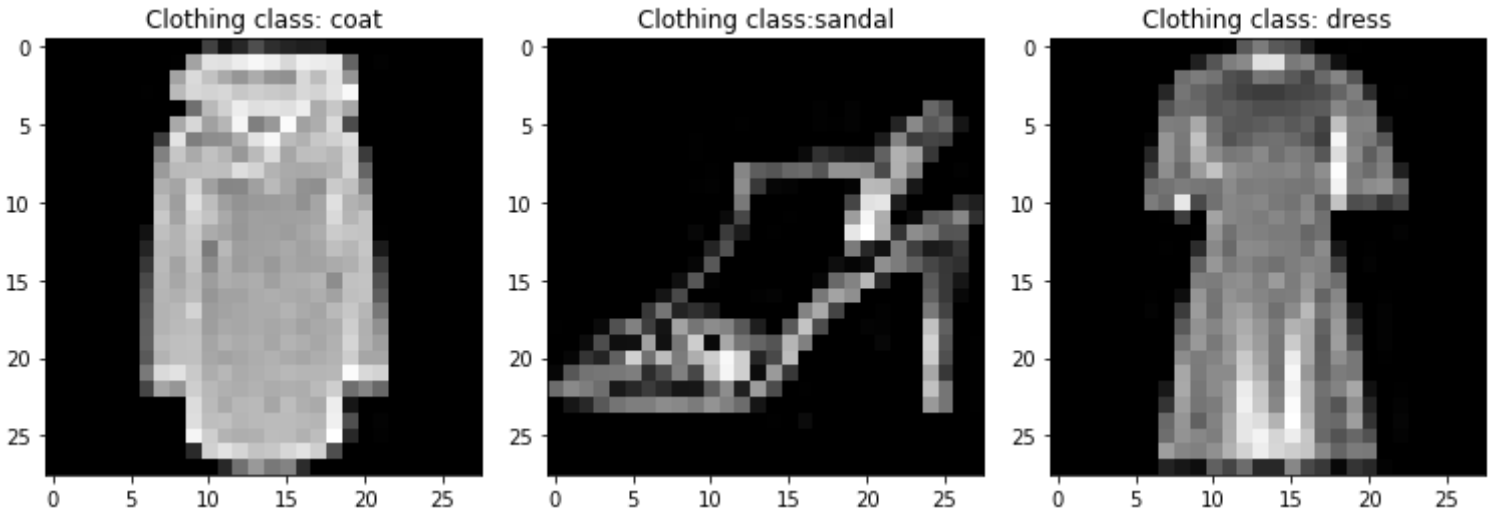
1 https://pjreddie.com/projects/mnist-in-csv/

Student Name: Bérénice Le Glouanec

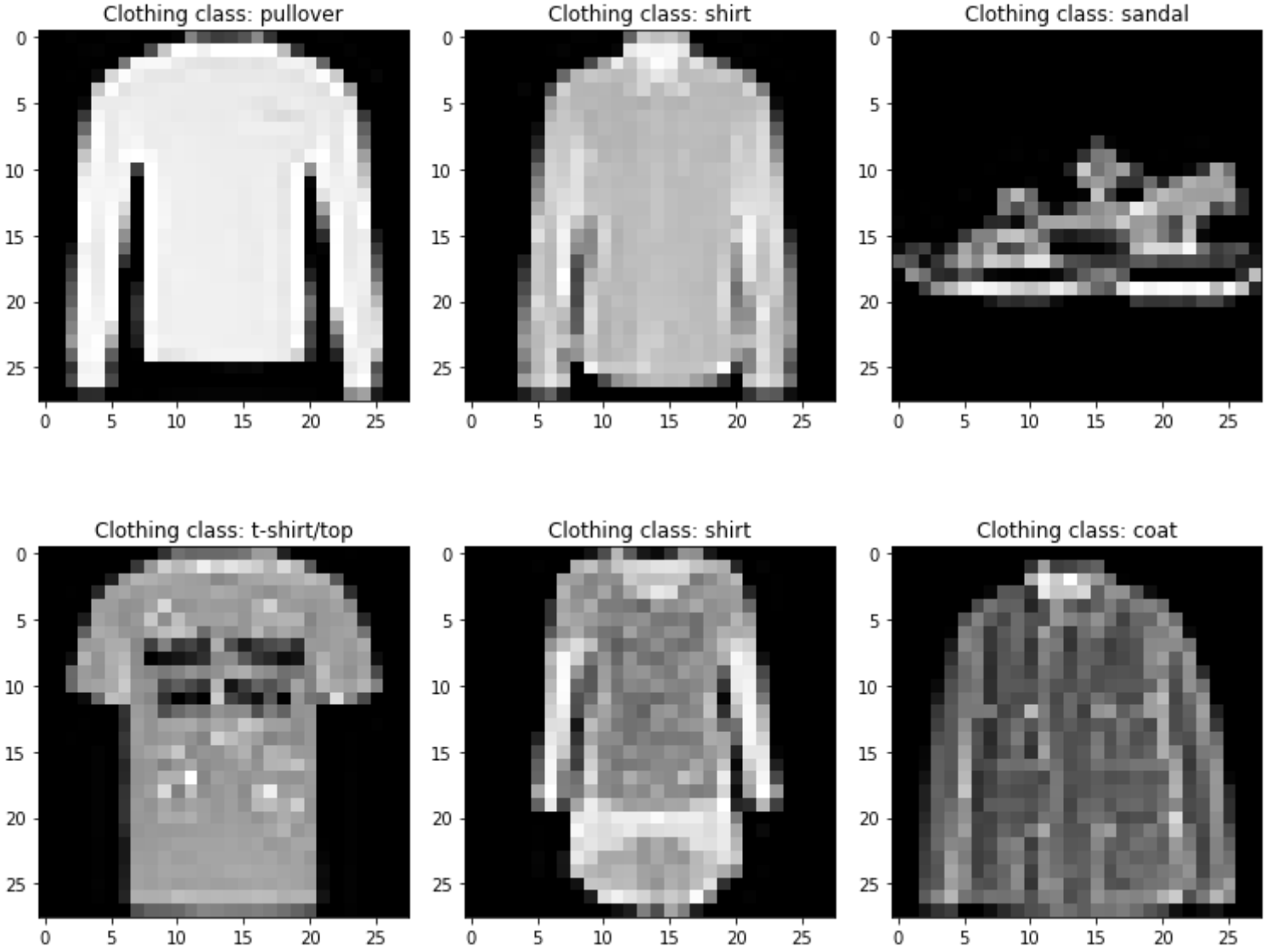


According to our results, there is no bias in any set, each label has the same chances to be correctly identified.

As it is important to understand what data we are working with, we reconstruct random images from the training set. This visualization helps to determine the visual quality of images (e.g. noisy images because of a bad quality that makes the object on an image impossible to be detected) and if there is any wrongly labeled instance (e.g. a picture of a shirt is labeled as a dress).



Student Name: Bérénice Le Glouanec



This small sample tends to indicate that the images are less likely to contain noise (neutral background, centered object, gray scaled …) or to be mislabeled. Additionally, as explained by the creators of Fashion-MNIST, the images share a standard format, size and quality. In this situation, there is no need to determine a structured standard polygon to structure the dataset with a consistent and uniform format, neither to perform noise reduction.

For the purposes of this task, we do not need to apply feature selection for the constructed feature space as the dataset contains only 10 classes in total. As we can not extract some features as more important than others in this set of classes, we perform a multi-class classification. Our model is based on LeNet-5 architecture model, and consists of 5 convolutional layers and 3 fully connected layers. Our model has been created with these layers because is not a good idea to use too many fully connected layers as they consume a lot of memory. In terms of accuracy, convolutional layers are more efficient. Indeed, convolutional layers provide a meaningful, low-dimensional and more or less stable feature space.

Student Name: Bérénice Le Glouanec

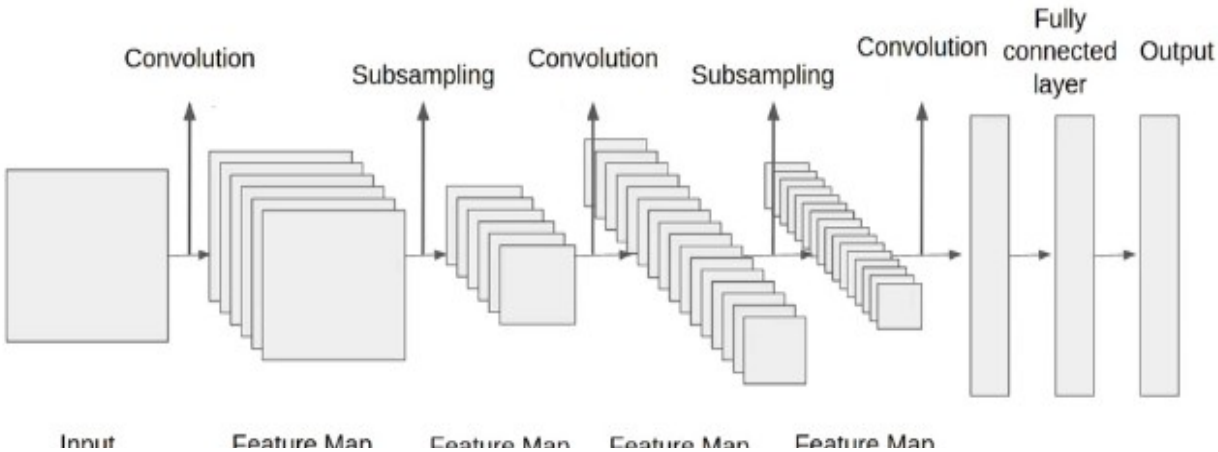


Illustration 1: Example of LeNet-5 implementation from Analytics Vidhya

Each layer contains and performs the following: convolutional layer, rectified linear activation function (ReLu), max pooling, batch normalization, dropout. We compute batch normalization after the activation layer (ReLu) and before dropout because it seems to be the convention according to most Machine Learning projects. We use small kernel because we do not need a big amount of pixels to recognize the object, therefore a small filter is sufficient. We add padding to keep a spacial size constant because it might improve performance as it adds rows and columns of zeroes. The two first fully connected layers are followed by a ReLu and dropout function. The last fully connected layer precedes log softmax function, which improves numerical performance over softmax function.-

Evaluation

Regarding the training and testing part, the hyper-parameters have been modified over time during our experiment, but for time-processing, and accuracy reasons, the following values are considered. We train our model on a batch size of 128, 10 epochs and a learning rate of 0.001. We choose the cross entropy loss as our optimizer loss function as it often used in multi-class classification, and we perform backwardization on the loss during the training process. The trained and tested models have been tested with same hyper-parameters and loss function. The autograd (gradient-based optimization) has been switched off as opposed to trained model.

Reducing learning rate from 0.0000001 to 0.001 (tested for each value multiplied by 10) really helped decreasing the loss. However, there is no improvement when reducing the learning rate to 0.01. The model has been evaluated on its accuracy on the training and testing sets. In order to do so, we sum all correct predicted output together and divide it by the total predictions. The accuracy is considered as better when it is closer to 100%. The losses of an epoch are considered as better when they reach 0.0 . In these results we can observe fairly good results from the beginning. As our trained and tested models both reach above 90% of accuracy, we can speculate that the CNN model is performing well.

Figure 1: Results Training Model

```
Training model ...

Total loss in epoch 0 is 0.43485391152693014
Total loss in epoch 1 is 0.30560425422720314
Total loss in epoch 2 is 0.27251004762868133
```

```
Total loss in epoch 3 is 0.24306826506342208
Total loss in epoch 4 is 0.22189195848095906
Total loss in epoch 5 is 0.19497087461226537
Total loss in epoch 6 is 0.17609799415000213
Total loss in epoch 7 is 0.17031409637506073
Total loss in epoch 8 is 0.14892758839706113
Total loss in epoch 9 is 0.13549517715043985
Train accuracy is  95.08166666666666
```

We can see the evolution of accuracy and loss at each epoch. As the results suggest it for epochs 6 and 7, the accuracy is exponential but would remain very similar if we were adding numerous following epochs.

Discussion and Future work

According to the results presented below, we can conclude that the model is performing very well, given that it obtains an accuracy of 95.08% on the training set and an accuracy of 92.37% when tested. These results also show that the images are consistent as there is no outlier instance, otherwise we would see one or several spikes on the following graphs.

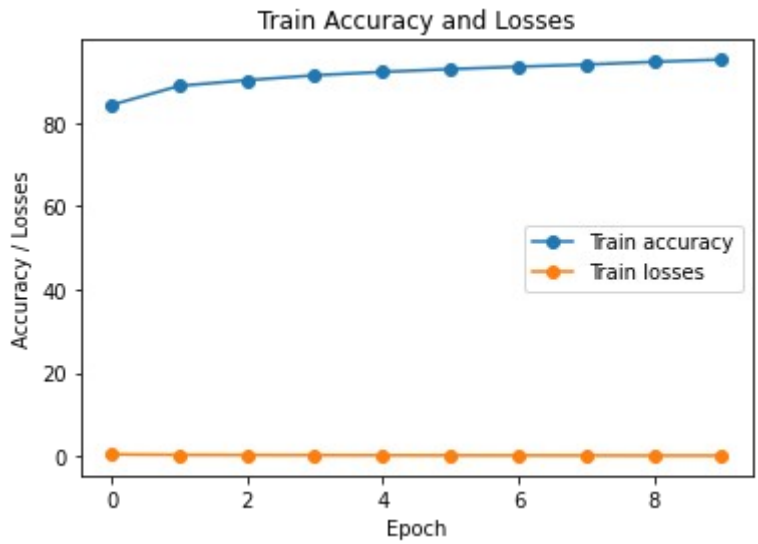


Figure 2: Train Accuracy and Losses

Student Name: Bérénice Le Glouanec

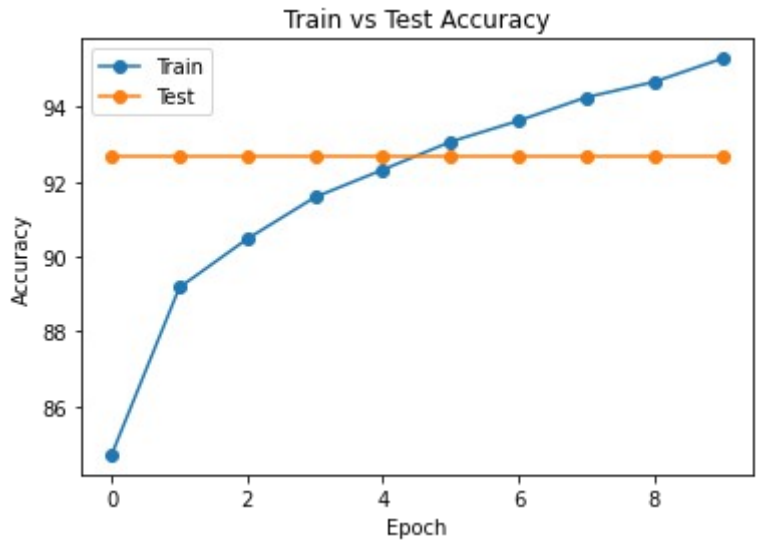


Figure 3: Train vs Test Accuracy

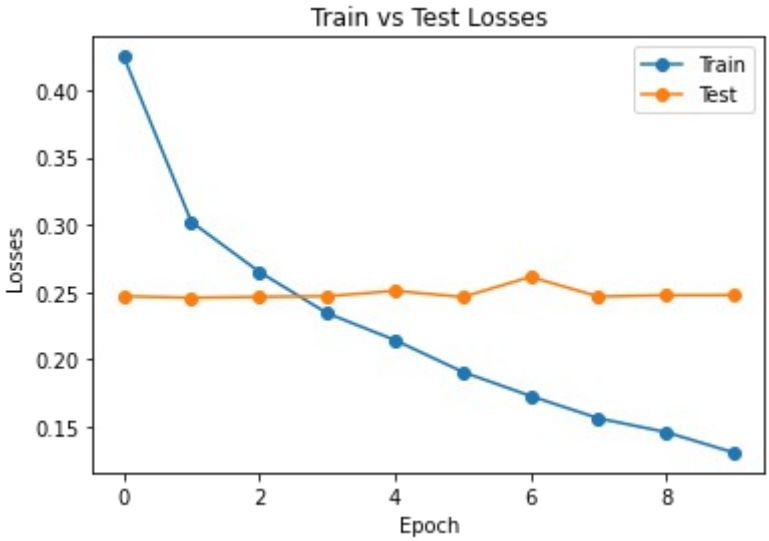
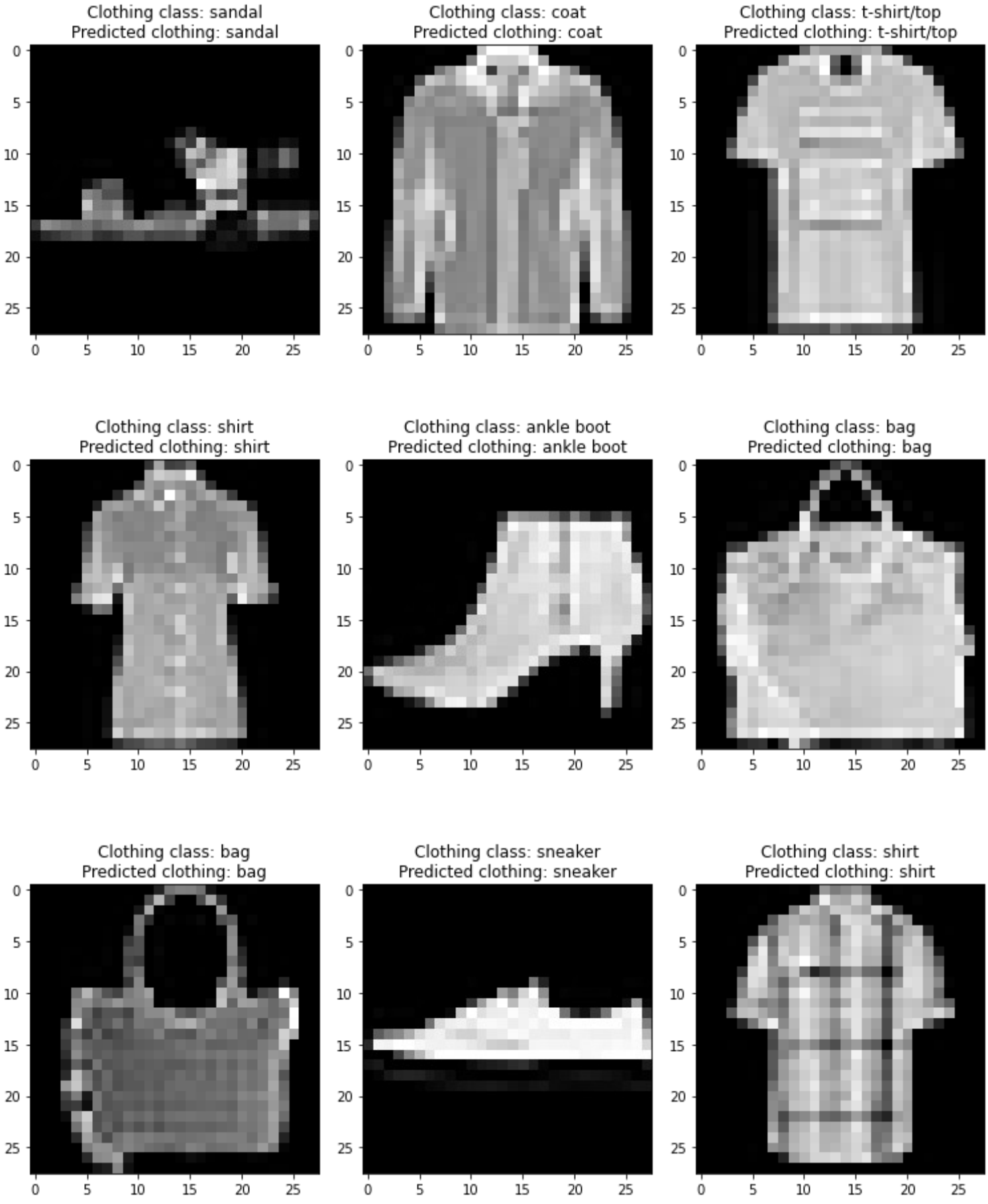


Figure 4: Train vs Test Losses

We represent a random sample of predicted instances and their original label in order to have an idea of the performances of our model. As it can be seen below, out of 9 examples, there is no incorrect prediction.

Student Name: Bérénice Le Glouanec



From these results, we can conclude that using several convolutional layer and fully connected layers was optimal. Furthermore, opting for a multi-class classification was adapted in this situation (e.g. having several but not too many classes without importance level or hierarchy among them).

However, considering our consistent and standard data, our model could be overfitting. It is performing well on this dataset, but it would be reasonable to question the model accuracy on even a slightly different dataset (for instance images with a white background, or smaller/ bigger objects).

In future work, it would be interesting to train the data with a greater batch size and epoch size. Depending on the results, we could then adapt the learning rate efficiently. Additionally, as we have seen that there was no significant noise in the dataset, it would be intriguing to make more complex the dataset by injecting noise in it so that it would be more efficient on the testing set. Indeed, previous work by X. Zhu and X. Wu (2004) has shown that training a model on a noisy dataset can give good results and perform very well.

In this experiment one metric tool has been performed over the data. However, if one wants other detailed results than accuracy, it would be advisable to compute some other metric tools like precision, recall or f1 score.

Conclusion and Summary

Along this experiment, we got acquainted with the Fashion-MNIST dataset in order to understand the data and formulate an efficient and machine-learning-related solution. We raised a classification prediction problem and organized our work accordingly.

We have implemented a Convolutional Neural Network based on LeNet-5 architecture model. We have checked whether our Fashion-MNIST dataset was potentially noisy or mislabeled. We have trained our model on specific hyper-parameters according to our needs. We have compared and evaluated our model based on its accuracy and losses during the training and testing part. We have visualized our data from the beginning to the final of our experiments in order to keep an eye on any disparity, and also to fully understand each step and outcomes of our process.

We have created a very good model, performing with about 95% accuracy on the training set, and about 92% accuracy when tested on an unknown set (testing set).

Finally, this work can be improved until the accuracy of each set reach 99%. To do so, some propositions have been raised, such as injecting noise in the training set and/or the testing set.

References

A guide to an efficient way to build neural network architectures- Part II: Hyper-parameter… published on Medium, website link, July 15 2021.

Convolutional Neural Network Champions -Part 1: LeNet-5 (TensorFlow 2.x) published on Medium, website link, June 1 2020.

Class Noise vs. Attribute Noise: A Quantitative Study published by X. Zhu and X. Wu, in the Artificial Intelligence Review journal, vol. 22, no. 3, pp. 177-210, November 01 2004.

Fashion-MNIST published on Github, website link, April 2017.

Student Name: Bérénice Le Glouanec

LeNet-5 CNN with Keras – 99.48% published on Kaggle, website link, February 19 2019.

Mnist in csv published on Pjreddie, website link.

The Architecture of Lenet-5 published on Analytics Vidhya, website link.

What Is Cross Entropy Loss? A Tutorial With Code published on W&B, website link, September 10 2021.