Advertima ML challenge

Completed by Oleg Borisov

In this challenge the goal is to create a NN architecture that is capable of classifying MNIST-fashion dataset images. According to the description of the dataset, many previously created Machine Learning architectures and methods perform worse on this dataset rather than on the "classical" MNIST.

The implementation was done in the PyTorch using Colab[1], as Colab provides for free the usage of GPU and hence allows to train models much faster than on CPU. Please note that as Mac user this was the only option for me to complete this challenge.
https://colab.research.google.com/drive/1C-_piqXlMMgxuE81RGo0Xm6j7qJlN82v

Train – Validation split

First of all, I made a split of training set to the: *training* and *validation* sets. After each training epoch the validation accuracy was checked and then the model with the **best** validation accuracy was considered to be the final model and the performance was checked on the *testing* dataset.

Architectures

Since this is an image recognition classification task, I decided to use Convolutional Neural Networks, I have experimented with 2 architectures that are presented below. [Check Figure 1 and Figure 2]

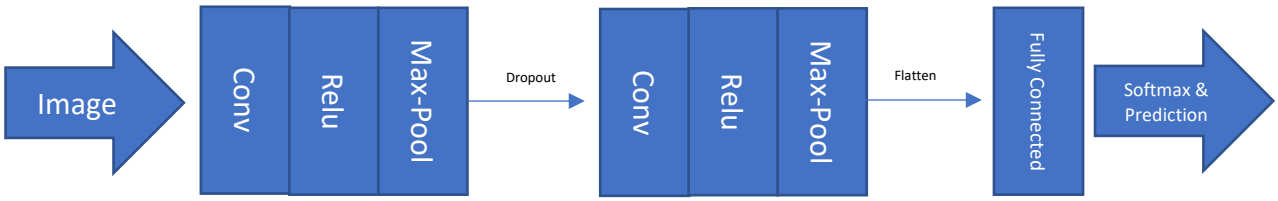

Figure 1: First Architecture, also referred as "*SimpleCNN*"

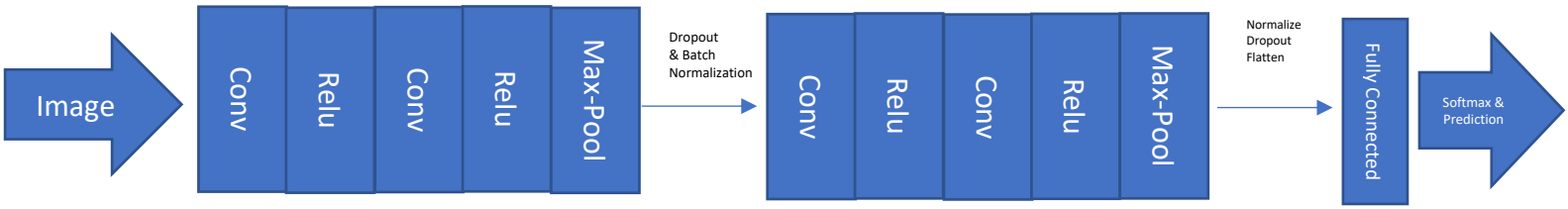

Figure 2: Second Architecture, referred as "*SecondCNN*"

After running training the above described architectures (given similar parameters), it was clear that the second architecture was able to obtain a better result over the same training time (91.42% test accuracy of the SimpleCNN vs. 92.2% of the second one). Therefore, I decided to continue exploring the second architecture. Figure 2 shows a sketch of the final architecture, during the experimentation I have changed some parameters and added Batch Normalization, therefore in the end the test accuracy has increased to 92.97%.

[1] https://colab.research.google.com/

Data Augmentation

In order to try to push the model to achieve even better results I have applied the Data Augmentation approach. In particular I have made horizontal image flips on random images (both original and flipped images were stored in the training set). Example is shown on Figure 3.

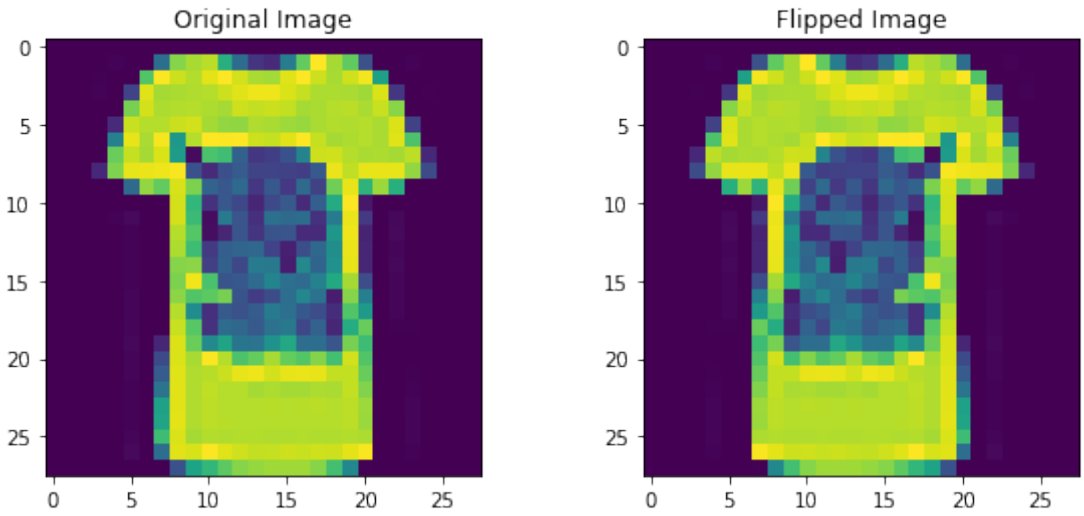

Figure 3: Horizontal Image Flip

Therefore, the training dataset has been increased and even better accuracy results have been obtained: 93.5% on test set (which took 1.5- 2h to train the network using 50 Epoch). I further have tried to improve the CNN architecture by increasing the number of Convolutional Channels, which unfortunately has not led to the great performance increase but the time to train the network increased dramatically. (Referred as *extended* dataset)

Loss functions

In the above examples I have tried Mean Squared Error (MSE) loss function, however the more commonly used loss function for CNNs is Cross-Entropy. By running the SecondCNN architecture with the Cross-Entropy loss function I have not achieved any improvements in the test accuracy.

Data Normalization

The other important thing to do is to normalize the data, i.e. given image x_i

$$\widehat{x_i} = \frac{x_i - \mu}{\sigma}$$

Where μ – is mean of images and σ – is standard deviation. Unfortunately, this approach has not helped either, and the test accuracy on the extended dataset was only 92.02%.

Dynamic Learning rate

Another method I have tried was Dynamic learning rate. The idea is that as more we train and approach the local minima, the initial learning rate might be too high to be able to approach the local minima and thus we will not be able to get the optimal result, therefore it could be a good idea to decrease the learning rate after some time.

Result

After experimenting with some different learning rates and epochs I was able to achieve the test set accuracy of 94.79%, which is close enough to the goal accuracy. The best accuracy was achieved by running the

SecondCNN architecture with 48 Conv filters in the first round and 96 Conv filters during the second round. It took 100 epochs to train the model where the first 50 had a learning rate of 0.001 and the other 50 had learning rate of 0.00025. The model was trained on the extended dataset, without using the data normalization. The downside of the training this model is that to train 100 epochs it takes 16.6 hours, however having more Conv filters has definitely helped to achieve a better model accuracy.