Image Classification on Fashion MNIST with Neural Networks

Papadopoulos Stefanos-Iordanis

1. Introduction

The current work attempts to recreate and evaluate various approaches based on Neural Networks (NN) for Image Classification and compare their performance against the Fashion MNIST dataset. The dataset was collected by Zalando's research team [1], a european e-commerce company, and has since become a commonly used 'baseline' dataset for benchmarking Image Classification algorithms, especially for initial testing and debugging.

To this end, various experiments including Multi-Layer Perceptrons (MLP), Convolutional Neural Networks (CNN), Pre-Trained Deep CNNs and Image Augmentation were performed and evaluated comparatively.

2. Methodology

This section briefly describes the general structure of the Fashion-MNIST dataset, the required preprocessing steps and thereafter analyses the architectures of the examined Neural Network algorithms.

2.1. Dataset description and Preprocessing

The Fashion MNIST dataset consists of 70.000 images of clothing items, categorised into 10 different classes. More specifically the classes are : T-shirt (0), Trouser (1), Pullover (2), Dress (3), Coat (4), Sandal (5), Shirt (6), Sneaker (7), Bag (8), Ankle Boot (9), as can be seen in Figure 1.

The size of the images is 28x28 pixels and have been transformed into grayscale matrices of values ranging from 0 to 255, on the black to white spectrum. Since Neural Networks require normalized values of 0 to 1, the matrices were ***MinMaxScaled*** from the initial range of (0 - 255) to (0 - 1). Furthermore, the total amount of items for each class is balanced, at 6000 items for each class, therefore no re-sampling balancing technique is required.

Finally, the dataset, as offered by Keras API, is already split into training and testing sets - 60 and 10 thousand respectively but the Training set was further split into Training (55.000) and Validation set (5.000) for reducing overfitting in the training process.

Figure 1

2.2. Algorithms

Four different approaches were selected and developed using Keras and Tensorflow and will be described in more detail in the following subsections. In all four architectures, the rectified linear unit (***RelU***) was selected as the activation function for the Hidden Layers, ***Softmax*** for the Output Layer, **Adam** (a variation of Stochastic Gradient Descent) as the optimizer and finally the ***Categorical Cross Entropy*** as the loss function (a probabilistic function, suitable for multi-class classification problems). This parameter combination has been shown to perform well in Computer Vision tasks [2] and thus, was used as a shortcut.

Furthermore, every model was trained with ***Reduced Learning*** and ***Early Stopping*** callbacks. The first reduces the learning rate once the training performance stagnates while the second monitors the validation loss function and ends the training process 'prematurely' if the loss function does not improve for a pre-defined number of iterations. In that case, the module can restore the best performing instance of the model.

2.2.1. Multi-Layer Perceptron

The Multi-Layer Perceptron (MLP) was selected as the baseline model of this study. MLPs are considered one of the more simple NN architectures consisting of one Input Layer, one or more Hidden Layers using a non-linear activation function and an Output Layer.

For the input layer the ***Flatten*** function was added in order to transform the input from a matrix to a flat sequence. Thereafter, the first experiment uses only one hidden layer of 256 nodes while the second adds a second hidden layer of 128 nodes. After each hidden layer a Dropout of 0.4 rate was added in an attempt to reduce overfitting. The ***Dropout layer*** forces the network to re-learn various independent representations by temporarily and randomly 'dropping' or ignoring the output of some neurons. It can especially be useful in smaller datasets such as Fashion-MNIST.

2.2.2. Convolutional Neural Networks

CNNs and their variations can be considered to be among the State-of-the-Art techniques in Computer Vision. The first layer requires a ***Conv2D*** layer that creates a feature map by sliding a convolution filter over the input image. Secondly, a ***MaxPooling2D*** layer for dimensionality reduction of each feature and lastly a ***Dropout Layer*** is again added to reduce the overfitting. Subsequently, the previous output must be flatten again so that it will be fed into two stacked Dense Layers, known as the ***fully connected layer***. This is the architecture of a simple "one - convolutional - layer" network. Such a model was developed with 32 filters while in later experiments a second convolutional layer is added with 64 and lastly a third convolutional layer of 128 filters.

2.2.3. Data Augmentation

Data augmentation is the technique of generating additional training data from the existing samples via random transformations. More specifically, in the context of image augmentation, existing images may be randomly rotated, shifted in height or width, zoomed, flipped (vertically or horizontally), resulting in realistically altered images. The goal of augmentation is the creation of a richer dataset that may help avoid overfitting by presenting the same items under slightly different conditions and characteristics. For this work, the ***Image Data Generator***, provided by the Keras API was selected, with a width and height shift of 0.1 and zoom of 0.08. The new augmented dataset was given to the same three CNN models described in the previous section.

2.2.4. Pre-trained Deep CNNs

Keras application API offers a VGG model pre-trained on the ImageNet dataset. VGG is a very deep CNN architecture consisting of 16 to 19 layers. The input images must be reshaped into 48*48 and 150*150 pixels for VGG16 and VGG19 respectively. The pre-trained model is used to extract features from the target dataset and the results are fed into a densely connected classifier for the final training.

3. Experiments

The experiment design of the four aforementioned approaches was :

1) MLP with one hidden layer (MLP1)
2) MLP with two hidden layers (MLP2)
3) CNN with one hidden layer (CNN1)
4) CNN with two hidden layers (CNN2)
5) CNN with three hidden layers (CNN3)
6) Image augmentation + CNN1 (A-CNN1)
7) Image augmentation + CNN2 (A-CNN2)
8) Image augmentation + CNN3 (A-CNN3)
9) VGG16 pre-trained on ImageNet
10) VGG19 pre-trained on ImageNet

All models were trained on the same training set, of 55.000 samples, validated on 5.000 samples and tested on the same 10.000 samples.

Generally used metrics for evaluating supervised classification tasks include ***Accuracy, Precision, Recall*** and ***F1 score*** displaying different relationships between correctly and incorrectly predicted classes of both positive and negative cases. All four evaluation metrics were utilized but due to the dataset being totally class balanced, no significant deviation was observed. Hence for practical reason only the mean ***Accuracy score*** will be presented and discussed. In this context, accuracy is defined as the summation of all True Positives with all True Negatives divided by the total population.

Furthermore, the Accuracy of all three phases, training, validation and testing, are presented for more precise evaluation and the history of training epochs for both accuracy and training loss were plotted to understand the progression of the training process, as can indicavelly be seen in Figure 2.

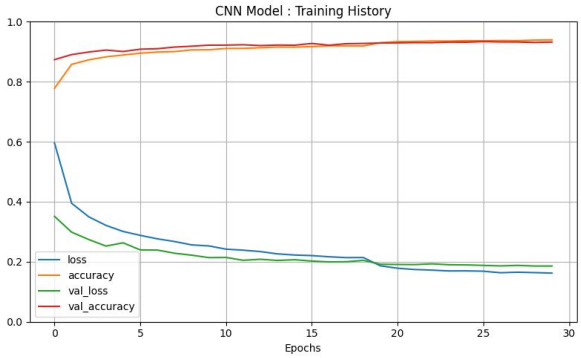

Figure 2.Training History of CNN3

4. Results

Based on the visualised results of Figure 3, the MLP models perform impressively well considering their relatively simplistic architecture. However, one notable issue is their overfitting on the training set, especially in the case of MLP1 where there is a 9.89% difference between training and testing accuracy.

More expectedly, the three CNN models out-performed the MLP, with the CNN3 scoring at 92.82%, and with lower relative rates of overfitting.

Surprisingly, Image Augmentation was not able to improve the predictive performance of CNNs even though they were the only model where overfitting was not an issue. Consistently, their training and validation accuracy scores are slightly lower than the testing accuracy.

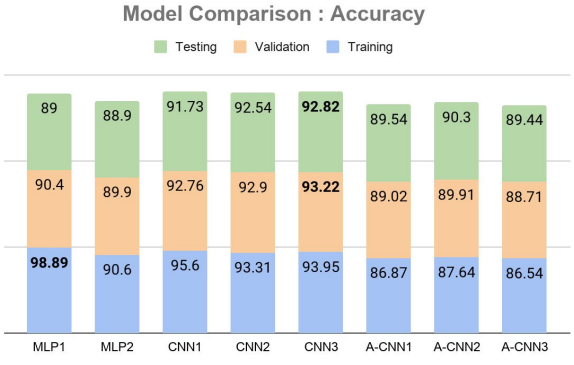

Figure 3

Finally, the VGG models are not presented in the plotted figure, since VGG16 showed a very poor performance of 28.08% and VGG19 was not able to complete training due to memory overload.

5. Conclusions

The restrained performance of Image Augmentation and the very poor performance of VGG16 can most probably be attributed to the lack of hyper-parameter tuning and experimentation with different Loss functions and Optimizers. It is likely that under different circumstances, these models would perform significantly better, as shown in many cases. [3] The lack of such techniques was due to technical limitations and no access to more powerful and distributed computational systems.

Apart from hyper-parameter tuning and experimenting with various Loss functions, Optimizers etc it may have been a worthwhile endeavor to experiment with semi-supervised Generative Adversarial Networks modified for image classification and other state-of-the-art methods such as Differentiable architecture search (DARTS) [4] and Mixed Sample Data Augmentation [5] which are two highest scoring methods in 'Paperswithcode' for benchmarking the Fashion-MNIST dataset.

6. References

[1] Xiao, Han, Kashif Rasul, and Roland Vollgraf. "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms." arXiv preprint arXiv:1708.07747 (2017).

[2] Zaheer, Raniah, and Humera Shaziya. "GPU-based empirical evaluation of activation functions in convolutional neural networks." 2018 2nd International Conference on Inventive Systems and Control (ICISC). IEEE, 2018.

[3] Perez, Luis, and Jason Wang. "The effectiveness of data augmentation in image classification using deep learning." arXiv preprint arXiv:1712.04621 (2017).

[4] Tanveer, Muhammad Suhaib, Muhammad Umar Karim Khan, and Chong-Min Kyung. "Fine-Tuning DARTS for Image Classification." arXiv preprint arXiv:2006.09042 (2020).

[5] Harris, Ethan, et al. "Fmix: Enhancing mixed sample data augmentation." arXiv preprint arXiv:2002.12047 (2020).