Machine and Deep Learning exploration of the Fashion-MNIST dataset

Simone De Renzis
simone.derenzis@studenti.unipd.it

1. Introduction

The aim of this report is to assess the performance of various classical Machine and Deep learning algorithms on the Fashion-MNIST dataset. Starting from basic models and arriving to the more complex ones, every algorithm is tuned with a hyperparameters grid search in order to get a suitable optimal model to compare with the others. To do this, a first exhaustive grid search is performed on PCA compressed data, in order to estimate in a faster way a subset of hyperparameters for every model: from these, another search is performed, this time on full resolution data, to obtain the best models. Moreover, different preprocessing and data augmentation techniques are explored to see if the performance can be improved.

Of all the models, the Convolutional Neural Network performed better and was the only one to score over 90% accuracy on the test set. Behind it the fully connected Neural Network. Surprisingly, Neural Network which images had been processed with HOG transformation and data augmentation (horizontal reflection) performed worse than the regular Neural Network. SVM performed nicely and just behind the Neural Network. On the bottom of the score, simpler models like Random Forest, KNN, Logistic Regression, Decision Tree and Perceptron, which however, all achieved over 80% of accuracy.

2. Dataset

The Fashion-MNIST dataset is a selection of 70 000 28x28 gray-scale images of fashion products from the Zalando database. With 60 000 images for the training set and 10 000 for the test set distributed among 10 classes, it resembles the structure of famous MNIST dataset that is often used as a benchmark in computer vision classification tasks. On regular MNIST it is easy to reach up to 97% of accuracy even with basic models: the Fashion-MNIST is harder, making it useful and more reliable in benchmarking advanced machine learning models.

Fashion samples belonging to different classes share some similar features and are indeed difficult to classify. UMAP (3) is a dimensionality reduction algorithm that provides very good visualization of data by projecting it in lower dimension. In fig. 2, UMAP has been used to plot the Fashion MNIST dataset (training set) in a 2d plot. Some classes are easily separated from the others, for example “Trouser” and “Bag”. “Sandal”, “Ankle boot” and “Sneaker” belong to the same cluster (they are indeed similar) but can be easily told apart. This does not apply for “T-shirt/top”, “Dress”, “Pullover”, “Shirt” which are mixed in the sample blob of points.

Figure 1: Samples from the Fashion-MNIST dataset

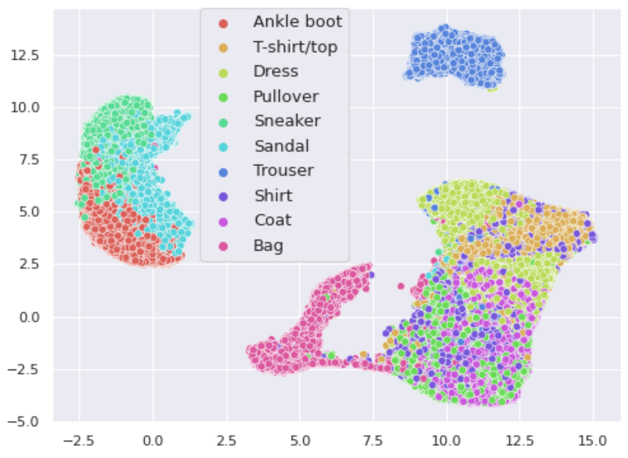

Figure 2: Visualization of the Fashion-MNIST dataset in a 2d plotting with UMAP, a dimensionality reduction technique. In this figure, the algorithm is run with number of neighbors parameter set to 50.

3. Models and preprocessing

3.1. Scaling

The images pixels, which range in $[0, 255]$ have been scaled so that they range in $[0, 1]$, simply by dividing every pixel value by 255. This makes algorithm (especially neural networks) compute faster and converge easier.

3.2. PCA compression and sampling

To find the best set of hyperparameters, every algorithm is run through a grid search and the best one is selected on a 5-fold cross validation. This operation is computationally expensive: to make it faster, a first exhaustive grid search is operated on PCA compressed images (fig 3), in order to get an estimation of how the hyperparameters range across the possible values. A subset of these are then chosen to be used on the full resolution images. Principal Component Analysis (PCA) is a technique based on finding the direction of maximum variance in the data to extract relevant information from it, and to reduce its dimensionality (4).

Along with PCA, the training set, which counts 60 000 images, has been reduced by selecting 2 000 samples out of each class, for a total of 20 000 images. In this way the dataset is reduced to $1/3$ of its original size, contributing to save computational time while keeping the classes balanced.

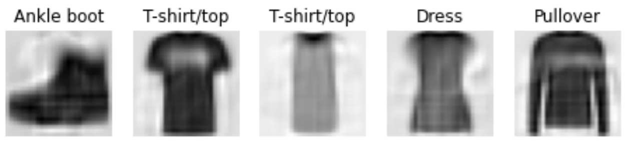

Figure 3: Samples of reconstructed images after being processed with PCA and reduced to 50 dimensions. Starting from 784 dimensions (flattened 28x28), the reduction is strong but the images are still recognizable even if some details have faded out.

3.3. Grid Search

In this section, the results for the grid search on full resolution images is shown. The hyperparameters are chosen with grid search on PCA compressed images (not shown here) and used to identify the two or three best models for each algorithm. For some models a short comment will pinpoint particular aspects about it.

The Perceptron (table 1) is the simpler model and is not expected to reach high accuracy. Regularization terms are used to keep overfitting controlled, and alpha is a constant that multiplies the regularization term.

Params	**Train acc**	**Valid acc**
'alpha': 1e-06, 'penalty': 'l2'	0.8261	0.8116
'alpha': 1e-05, 'penalty': 'l1'	0.8196	0.8021
'alpha': 1e-06, 'penalty': 'l1'	0.8148	0.7977

Table 1: Top three models for Perceptron.

Params	**Train acc**	**Valid acc**
'criterion': 'entropy', 'depth': 13	0.9173	0.8166
'criterion': 'entropy', 'depth': 12	0.8974	0.8155
'criterion': 'gini', 'depth': 13:	0.9122	0.8146

Table 2: Top three models for Decision Tree.

Params	**Train**	**Valid**
'C': 5, 'fit_intercept': True	0.8873	0.8468
'C': 10, 'fit_intercept': True	0.8875	0.8459
'C': 15, 'fit_intercept': True	0.8877	0.8458

Table 3: Top three models for Logistic Regression.

Params	**Train**	**Valid**
'n_neighbors': 6	0.8910	0.8550
'n_neighbors': 8	0.8820	0.8547
'n_neighbors': 5	0.8966	0.8540

Table 4: Top three models for KNN.

Random Forests (table 5) obtained pretty high accuracy scores, but having deep trees and a large number of estimators lead to strong overfitting. Further increasing these parameters would degrade the performance.

Params	**Train**	**Valid**
'max_depth': None, 'n_estimators': 55	0.9999	0.881083
'max_depth': 25, 'n_estimators': 55	0.9999	0.880633
'max_depth': None, 'n_estimators': 50	0.9999	0.880350

Table 5: Top three models for Random Forest.

SVMs (table 6) reached a high level of accuracy on the validation set while keeping overfitting under control. The one with the RBF kernel seems to be the best model between the ones tested so far.

Params	**Train**	**Valid**
'C': 1, 'kernel': 'rbf'	0.9111	0.8895
'C': 1, 'kernel': 'linear'	0.9070	0.8553

Table 6: Top two models for Support Vector Classifier.

3.4. Neural Networks

A similar approach has been applied to neural network models: a grid search over the number of hidden layers, the number of neurons per layer and the batch size allowed to identify some suitable models, reported in table 7.

Params	Train	Valid
'hidden_layers': 2, 'units_layer': 100	0.9356	0.8907
'hidden_layers': 3, 'units_layer': 150	0.9227	0.8896
'hidden_layers': 1, 'units_layer': 200	0.9317	0.8895

Table 7: Top three models for Neural Network. In all cases, a batch size of 20 has been used since it showed much better performance than bigger batch sizes like 200 and 500.

3.4.1 HOG transformation

HOG transformation is an image processing technique based on histogram of oriented gradients (HOG) to perform object recognition (2). In table 4, a visualization of how HOG transformed images look like.

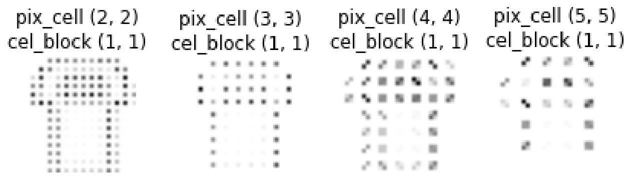

Figure 4: The result of HOG transformation in a "T-shirt" sample at different levels of pixel per cells: the value $(2, 2)$ preserves many of the original details and actually produces a valuable representation of the image: this value will be used to preprocess data in Neural Networks.

3.4.2 Data augmentation

Data augmentation is the process of creating synthetic sample data from the original dataset, to improve the performance of predictors. Given a sample image, possible operations to create new data are rotations, crop, zoom, reflection (horizontally and vertically) and translation. For the Fashion-MNIST dataset, since the images are centered, all at the same zoom level and at the same orientation, the only transformation that I applied was horizontal reflection.

Both HOG transformed and augmented data have been fed to Neural Networks with the same architectures of the ones in table 7.

3.4.3 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a special kind of Neural Network in which some of its layers are convolutional: the aim of the training procedure is to learn *kernels* (or *filters*), which are grids of parameters that, when convoluted on the input image, highlight important features on it (1). It's a very powerful and popular model especially for computer vision tasks.

For Fashion-MNIST classification I set up a 2 layers CNN: one is convolutional with 60 3x3 filters (followed by a max-pooling layer of 2x2), the second one is a fully connected layer with 200 neurons.

4. Results

From the results of grid search, the most performing models for every algorithm has been chosen and compared to the others based on the validation accuracy. From table 8, the CNN appears to be the best model, so it's the one that I would choose to perform classification task on the Fashion-MNIST dataset. Surprisingly, Neural Networks with preprocessed data (HOG transformed and augmented) performed slightly worse than Neural Network with no preprocessed data. Probably, different architectural choices had to be taken in these models to exploit the full potential of the preprocessing. SVMs scored nicely and just below Neural Networks. Simpler models placed on the bottom of the table, however Random Forests and KNN still achieved decent results.

These results are confirmed by the test accuracy obtained by the same models in the test set: the leaderboard of table 9 maintains the same order, and the test scores are only slightly worse than those obtained from the validation set.

Model	Validation score
CNN	0.9130
NN	0.8907
SVC	0.8895
NN with data augmentation	0.8872
NN with HOG transformation	0.8838
Random Forest	0.8811
KNN	0.8550
Logistic Regression	0.8468
Decision Tree	0.8166
Perceptron	0.8116

Table 8: Comparison of the best models for each algorithm on validation accuracy.

Model	Test score
CNN	0.9077
NN	0.8838
SVC	0.8828
NN with data augmentation	0.8771
NN with HOG transformation	0.8759
Random Forest	0.8723
KNN	0.8544
Logistic Regression	0.8389
Decision Tree	0.8131
Perceptron	0.8063

Table 9: Comparison of the best models for each algorithm on test accuracy.

References

[1] Saad Albawi, Tareq Abed Mohammed, and Saad ALZAWI. Understanding of a convolutional neural network, 08 2017.

[2] Kh Tohidul Islam, Ram Raj, and Abdullah Al-Murad. Performance of svm, cnn, and ann with bow, hog, and image pixels in face recognition, 12 2017.

[3] James Melville Leland McInnes, John Healy. Umap: Uniform manifold approximation and projection for dimension reduction, 2018. arXiv:1802.03426.

[4] Jonathon Shlens. A tutorial on principal component analysis, 2014. arXiv:1404.1100.