1. **Introduction**

We intend Fashion-MNIST to serve as a direct drop-in replacement for the original MNIST dataset [1] for benchmarking machine learning algorithms.

Fashion-MNIST project report

It shares the same image size and structure of training and testing splits, but instead of classifying the first 10 numbers, our objective is to classify 10 different types of clothes, which we did with an accuracy of 0.8737 by using a Neural Network.

2. **Dataset**

Fashion-MNIST is a dataset of Zalando's article images consisting of a training set of 60,000 examples and a test set of 10,000 examples. These datasets were loaded from https://www.math.unipd.it/~dasan/ under the variables X_train_val, Y_train_val, X_test, Y_test.

Each example is a 28x28 grayscale image, already flatten it into a vector and associated with a label from 10 classes. In Fig.1 [2] there is an example of how the data looks.

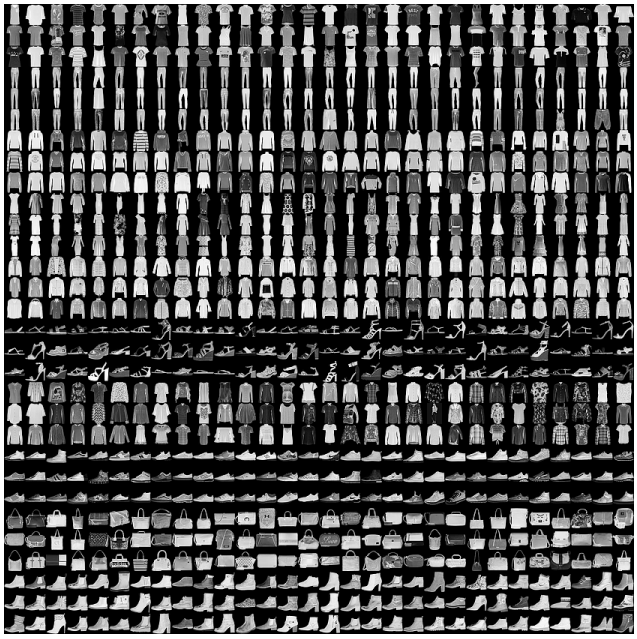


Fig. 1: Example of how the data looks (each class takes three-rows)

These classes are:

0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

However, to speed up the learning and testing operations, we initially decided to work with only 10% of the training

data. Other preprocessing techniques include creating a validation and test set from the datasets (0.2 split each) and using a StandardScaler to improve the performance of the Neural Network.

3. Method

As mentioned before, we decided to train and test our models firstly on the reduced dataset, and only after finishing all the setups, on the whole dataset.
The models used were: DecisionTreeClassifier, RandomForestClassifier, SVC and a Neural Network.

We decided to keep for the final testing phase with the full dataset only the model with the best Validation accuracy. To the first three models we also applied GridSearchCV to get the best parameters.

For the Neural Network we instead decided to use a trial-and-error approach to fix the overfitting problem we quickly discovered. Firstly, we tried a simple network with 256 hidden nodes and early stopping, which led to very underwhelmingly results, improved only by using StandardScaler to scale our training, validation and test sets. Then, to reduce overfitting, we added the Dropout method and the l2 Regularization [3], while also reducing the hidden layer to 64 nodes.

After completing the setup and testing phase, SVC ended up being our best classifier. We then performed some data exploration through a Confusion Matrix to see which classes were more likely to be misclassified and we found that most of the errors came from the misclassification of class 6 (shirt) with class 0 (T-shirt) and class 2 (dress).

We deemed this error understandable since these types of clothing have a very similar silhouette.

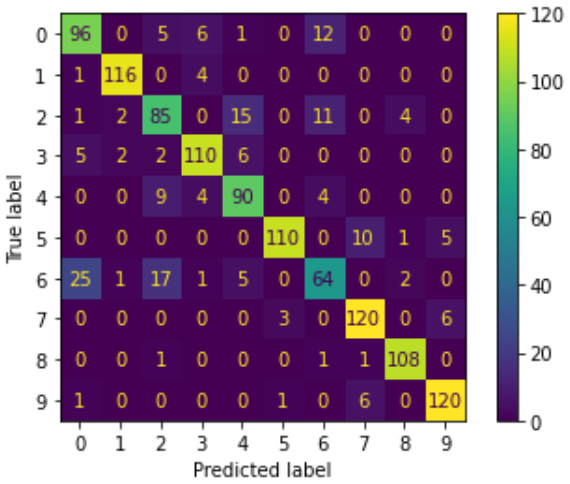


Fig. 2: Confusion Matrix of SVC on the test set

Nevertheless, we tried some oversampling techniques to improve the performances of these three classes, but the improvements were quite meaningless, so we scrapped them.

Finally, we trained and tested SVC on the complete dataset. Unfortunately we found out that SVM is not suitable for classification of large data sets, because the training complexity of SVM is highly dependent on the size of data set. Therefore we decided to use instead our second best model, the Neural Network.

4. Experiments

Results on 10% dataset, obtained by using the function train_test_split with parameter test_size=0.9.
Moreover, we used the parameter stratify=Y_train_val to keep the classes balanced.
From here we then split the dataset in train+val/test set and in train/validation set (0.2 each).

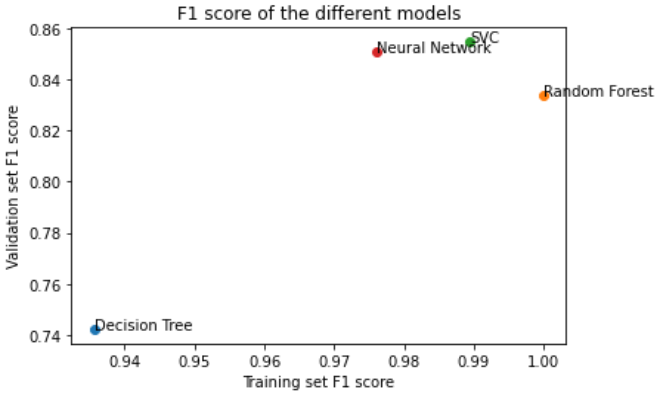


Fig. 3: F1 score of our models

Decision Tree: Best parameters were 'criterion'= 'entropy' and 'max_depth'=10.
Training accuracy=0.9356, Validation accuracy=0.7423.

Random Forest: Best parameters were 'criterion'='entropy', 'max_depth'=25 and 'n_estimators'=50.
Training accuracy=1.0000, Validation accuracy=0.8335.

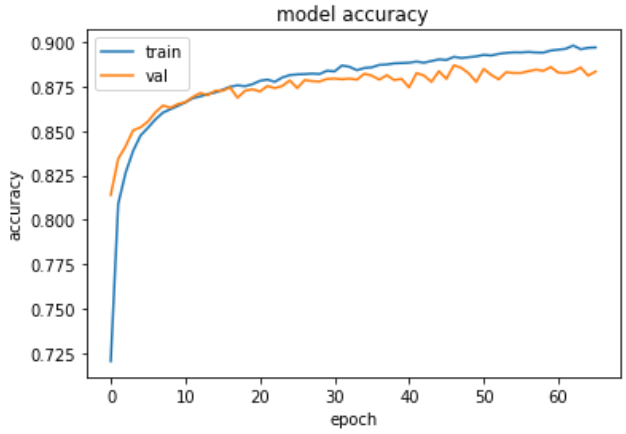
SVC: Best Parameters were 'C': 10 and 'kernel': 'rbf'.
Training accuracy= 0.9894, Validation accuracy=0.8546.

Neural Network: After several attempts we went from Training accuracy=0.0181 and Validation accuracy= 0.0185 to Training accuracy=0.9762 and Validation accuracy=0.8507. The results on the test set were instead: Loss=0.7010 and Test accuracy=0.8475

Since SVC was our best classifier, we tested it again on the test set obtaining Training accuracy=0.9628 and Test accuracy= 0.8450.

Results on the full dataset, using the train+val/test set and the train/validation set split (0.2 each).

As mentioned before, trying to model the full dataset with SVC ended up with a memory leak, so we used instead the Neural Network, whose results are the following:
Training accuracy=0.9021,Validation accuracy=0.8839.
Early stopping at epoch 00066.
The test results were instead:
Loss=0.4855, Test accuracy=0.8737,
Precision=0.8845, Recall=0.8841



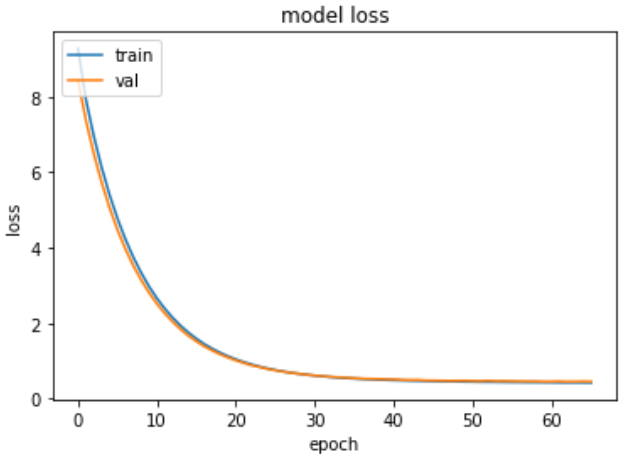


Fig. 4: Accuracy and model loss of the Neural Network on the full dataset (train and validation)

References

[1] THE MNIST DATABASE of handwritten digits Yann LeCun, Courant Institute, NYU, Corinna Cortes, Google Labs, New York Christopher J.C. Burges, Microsoft Research, Redmond, http://yann.lecun.com/exdb/mnist/

[2] Github, https://github.com/zalandoresearch/fashion-mnist

[3] An Overview of Regularization Techniques in Deep Learning (with Python code), Shubham.jain Jain, April 19, 2018, https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/