# Fashion-MNIST

Image classification with different types
of image transformation

Nicola Cassetta
nicola.cassetta@studenti.unipd.it

Andrea Di Trani
andrea.ditrani@studenti.unipd.it

## 1 Introduction

It is common knowledge that the more layers and neurons a neural networks has, the better it performs. However it is proven that a good image manipulation can bring better results together with less computational time and costs [1]. These are the assumptions of our work: we tested different types of image transformation and augmentation in order to exploit all the possible hidden information held by the Fashion-MINIST images. We found out that increasing the contrast and resampling the images, improve the performance of our models.

## 2 Dataset

The Fashion-MNIST dataset [2] is based on the assortment on Zalando's website. The original pictures are fed into a conversion pipeline, which results in a total on 70,000, 28x28 greyscale images, divided into 10 classes:

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

The pictures are divided into a training set of 60,000 examples and a test set of 10,000 examples. We further divided the training set in training and validation using 20% of the training set, resulting in a 70-20-10(%) split.

## 3 Method

As a first step we tested the vanilla version of the dataset with some of the most common classifiers, provided by the Sklearn package. In order:

- Decision Tree
- Random Forest
- Support Vector Machine (SVM)

During the tests we divided each pixel value by 255 modifying the range of those values to $[0, 1]$. All these models have been trained on the training set, and evaluated on the validation set.

After, we tested a very simple MLP (two hidden layer with 30, 20 neurons), in order to have a general idea of the results.

Next, we wanted to discover the best parameters for the classifiers, thus we improved our base test with a grid search. We tested a wide range of parameters for each model:

- Decision Tree:
  - Max depth: None, 3, 10, 20, 50
  - Min samples split: 2, 10, 50
  - Min samples leaf: 1, 2, 5, 10, 50
- Random Forest:
  - Max depth: None, 3, 5, 20, 50
  - Min samples split: 2, 10, 50
  - N° estimators: 5, 10, 20, 50
- Support Vector Machine (SVM):
  - C: 0.1, 0.5, 1, 5, 10
  - Kernel: rbf, poly, sigmoid

resulting on a solid 81%+ accuracy on the validation split among all the models, with a peak of 90% achieved by the SVM with the "rbf" kernel and C=10.

### 3.1 Image transformation and Data augmentation

Since the Fashion-MNIST is one of the most used dataset in the field of image classification, and due to the wide number of research papers available with great results, we decided, rather than testing only different models, to focus our attention on the exploitable hidden information held by the images themselves. After an exhaustive research, we selected different type of image transformation and data augmentation:

- Image transformation:
  - Contrast augmentation
  - Conversion from greyscale to black and white image
  - CLAHE (Contrast Limited Adaptive Histogram Equalization) from OpenCV
- Upscaling: we used the Pillow's Filter class to apply the Lanczos upscaling method, increasing the size of the images up to $64 \times 64$ [3]
- Data augmentation:
  - Clockwise 90° rotation
  - Flip to right
  - Flip to bottom

The idea behind the choice of these transformation rely on a handmade data analysis phase: during this step we visualized some of the samples and made some hypothesis:

- A more defined image could be more expressive, and less prone to be miss-classified. We tested all the transformation mentioned above and we found out that the best results were given using a higher contrast.

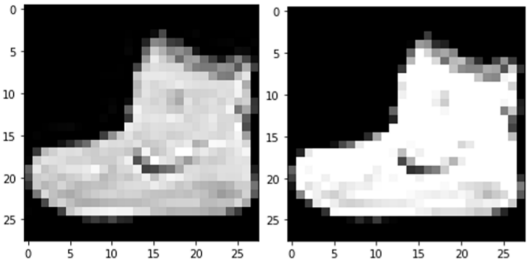

Figure 1: (Starting from left) a random image from the training set pre and post contrast augmentation (+50%). It is possible to see how the greyish alone in the first image, is not present in the second one.

- During the first test phase, we noticed from the Confusion Matrices, that many samples of certain classes were missclassified. The most problematic one was the Shirt class, often confused with T-shirt

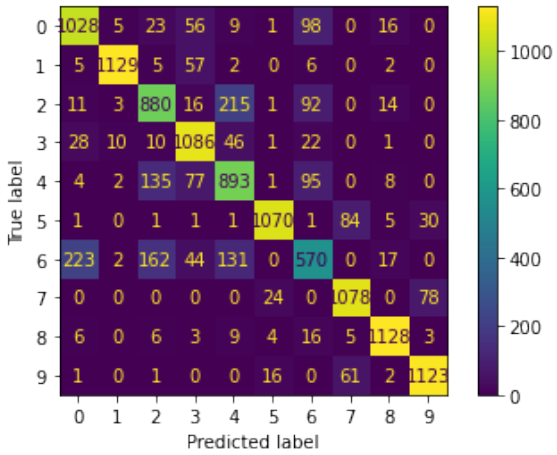


Figure 2: Confusion Matrix of a standard Random Forest.

In order to address this problem, we first visualized the missclassified samples, and we hypothesized that during the conversion pipeline, used to build the dataset, some important information were lost. For this reason we scheduled a test suite with the upscaled samples.

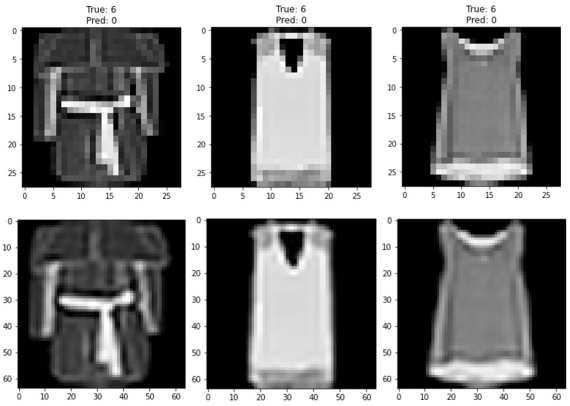


Figure 3: On the first line, the standard images 28x28. On the second line, the same images upscaled to 64x64 with the Lanczos method.

### 3.2 MLP architecture

In order to achieve the best results we planned an intensive search, with some custom functions, among different values for different hyperparameters, techniques and architectures. We now proceed to list some of the tested hyperparameters:

- Weight initializers: Random Normal, Random Uniform, Glorot Normal, Glorot Uniform
- Optimizers: SGD, Adam, Rmsprop
- Dropout: range of value between [0.2, 0.8] depending on the size of the network, as suggested in [4]
- Batch normalization
- N° of hidden layers and neurons: wide range of values, depending on the number of the features

## 4 Experimets

As mentioned in the section 3, we first tested the SVM, Random forest (RF) and Decision Tree (DT) on the vanilla dataset, and then we searched via grid search the best parameters. The highest values, among the best params combination, are summarized below.

**Random forest:** The best results were given by the combination of

- Critetion: Entropy
- Max depth: None
- Min samples split: 10
- N° of estimators: 50

| Val F1 | Val Acc | Test F1 | Test Acc |
|---|---|---|---|
| 0.876 | 0.878 | 0.870 | 0.872 |

**Decision Tree:** The best results were given by the combination of

- Critetion: Entropy
- Max depth: 20
- Min samples split: 10
- Min samples leaf: 50

| Val F1 | Val Acc | Test F1 | Test Acc |
|---|---|---|---|
| 0.813 | 0.814 | 0.807 | 0.808 |

**SVM:** The best results were given by the combination of

- C: 10
- Kernel: Rbf

| Val F1 | Val Acc | Test F1 | Test Acc |
|---|---|---|---|
| 0.901 | 0.901 | 0.896 | 0.896 |

**MLP two hidden layer:** The first hidden layer with 30 neurons, and the second one with 20. Trained for 100 epochs with 256 as batch size

- Optimizer: SGD
- Act function: ReLu
- Weigths initializer: Random normal

| Last | | Best | |
|---|---|---|---|
| Val Acc | Val Loss | Val Acc | Val Loss |
| 0.876 | 0.340 | 0.884 | 0.322 |

### 4.1 Dropout, Batch normalization and improvement

The results of the MLP were promising. However we toughed that Dropout [5] and Batch Norm [6] could both improve them while reducing the noise on the learning curves.

**MLP with Dropout (MLP_D):** Two hidden layer with 30,20 units

- Optimizer: SGD(lr=0.2)
- Weigths initializer: Glorot normal
- Act function: ReLu
- Dropout: 0.2

| Last | | Best | | |
|---|---|---|---|---|
| Val Acc | Val Loss | Acc | Val Acc | Val Loss |
| 0.888 | 0.315 | 0.890 | 0.895 | 0.307 |

**MLP with Dropout and Batch normalization (MLP_DB):** Two hidden layer with 60,40 units

- Optimizer: Adam
- Weigths initializer: Random uniform
- Act function: ReLu
- Dropout: 0.6

| Last | | Best | | |
|---|---|---|---|---|
| Val Acc | Val Loss | Acc | Val Acc | Val Loss |
| 0.897 | 0.425 | 0.909 | 0.903 | 0.383 |

**MLP with Dropout, Batch normalization, Upscale and Contrast (MLP_DBUC):** Four hidden layer with 2048, 1024, 256, 30 units

- Optimizer: SGD
- Weigths initializer: Random normal
- Act function: ReLu
- Dropout: 0.5

| Last | | Best | | |
|---|---|---|---|---|
| Val Acc | Val Loss | Acc | Val Acc | Val Loss |
| 0.898 | 0.345 | 0.931 | 0.898 | 0.345 |

**MLP with Dropout, Batch normalization and Data augmentation (MLP_DBA):** Three hidden layer with 256, 30, 20 units

- Optimizer: Adam(lr=0.01)
- Weigths initializer: Random normal
- Act function: ReLu
- Dropout: 0.4

| Last | | Best | | |
|---|---|---|---|---|
| Val Acc | Val Loss | Acc | Val Acc | Val Loss |
| 0.891 | 0.484 | 0.900 | 0.898 | 0.427 |

### 4.2 K-Fold cross-validation

As a last step we tested our best models in a 10-fold cross-validation. The results ore obtained computing the mean among all the folds.

**MLP with Dropout and Batch normalization (K_MLP_DB):** two hidden layer with 30, 20 units

- Optimizer: Adam
- Weigths initializer: Random uniform
- Act function: ReLu
- Dropout: 0.2

| Val Acc | Val Loss | Dev |
|---|---|---|
| 0.898 | 0.461 | 0.28 |

**MLP with Dropout, Batch normalization and Contrast (K_MLP_DBC):** two hidden layer with 30, 20 units

- Optimizer: Adam
- Weigths initializer: Random uniform
- Act function: ReLu
- Dropout: 0.2

| Val Acc | Val Loss | Dev |
|---|---|---|
| 0.895 | 0.461 | 0.48 |

**MLP with Dropout, Batch normalization, Contrast and Data Augmentation (K_MLP_DBCA):** two hidden layer with 30, 20 units

- Optimizer: Adam
- Weigths initializer: Random uniform
- Act function: ReLu
- Dropout: 0.2

| Val Acc | Val Loss | Dev |
|---|---|---|
| 0.896 | 0.556 | 0.37 |

# 5 Conclusion and Results

We now presents the results on the test set of our best models.

| Model | Test Acc | Test Loss |
|---|---|---|
| MLP_ | 0.870 | 0.375 |
| MLP_D | 0.878 | 0.353 |
| MLP_DB | 0.889 | 0.441 |
| **MLP_DBUC** | **0.882** | **0.366** |
| MLP_DBA | 0.880 | 0.472 |
| K_MLP_DB | 0.891 | 0.480 |
| K_MLP_DBC | 0.890 | 0.470 |

From the exposed results it is possible to see that the combination of Dropout, Batch Normalization, Contrast augmentation and Upscaling gives the best performance. Even if the model MLP_DB presents a slightly higher result on the accuracy, its training curve presents some issues:

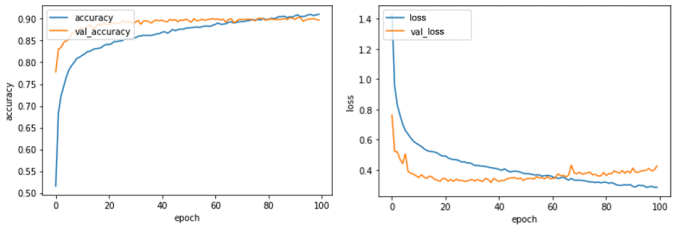


Figure 4: MLP_DB Training curves (Left: validation and training Accuracy, Right: validation and training Loss)

From this plot it is clear that the models tends to overfit, this behavior can be diagnosticated by the increasing validation loss while the training one keeps decresing.This did not happen with the model MLP_DBUC as can be seen in the plot below.

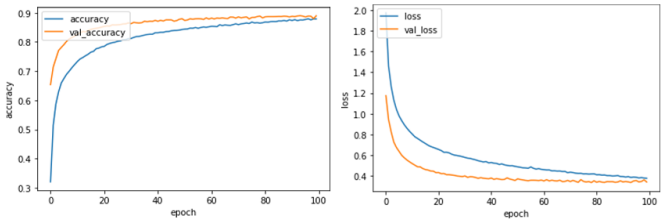


Figure 5: MLP_DBUC Training curves (Left: validation and training Accuracy, Right: validation and training Loss)

For technical reasons it was not possible to apply the K-Fold cross-validation to the upscaled, augmented dataset, for this reason we present the results on the _DB and _DBC models.
The results with the 10-Fold cross-validation are almost identical, this brought us to assume that the best results were given by the combination of all the exposed techniques, rather then using them separately.

# References

[1] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *arXiv preprint arXiv:1712.04621*, 2017.

[2] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for

benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[3] "Pillow documentation."

[4] C. Garbin, X. Zhu, and O. Marques, "Dropout vs. batch normalization: an empirical study of their impact to deep learning," *Multimedia Tools and Applications*, pp. 1–39, 2020.

[5] "Dropout: a simple way to prevent neural networks from overfitting,"

[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.