Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms

Han Xiao  
Zalando Research  
Mühlenstraße 25, 10243 Berlin  
han.xiao@zalando.de

Kashif Rasul  
Zalando Research  
Mühlenstraße 25, 10243 Berlin  
kashif.rasul@zalando.de

Roland Vollgraf  
Zalando Research  
Mühlenstraße 25, 10243 Berlin  
roland.vollgraf@zalando.de

Abstract

We present Fashion-MNIST, a new dataset comprising of $28 \times 28$ grayscale images of 70,000 fashion products from 10 categories, with 7,000 images per category. The training set has 60,000 images and the test set has 10,000 images. Fashion-MNIST is intended to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms, as it shares the same image size, data format and the structure of training and testing splits. The dataset is freely available at https://github.com/zalandoresearch/fashion-mnist.

1 Introduction

The MNIST dataset comprising of 10-class handwritten digits, was first introduced by LeCun et al. [1998] in 1998. At that time one could not have foreseen the stellar rise of deep learning techniques and their performance. Despite the fact that today deep learning can do so much the simple MNIST dataset has become the most widely used testbed in deep learning, surpassing CIFAR-10 [Krizhevsky and Hinton, 2009] and ImageNet [Deng et al., 2009] in its popularity via Google trends[1]. Despite its simplicity its usage does not seem to be decreasing despite calls for it in the deep learning community.

The reason MNIST is so popular has to do with its size, allowing deep learning researchers to quickly check and prototype their algorithms. This is also complemented by the fact that all machine learning libraries (e.g. scikit-learn) and deep learning frameworks (e.g. Tensorflow, Pytorch) provide helper functions and convenient examples that use MNIST out of the box.

Our aim with this work is to create a good benchmark dataset which has all the accessibility of MNIST, namely its small size, straightforward encoding and permissive license. We took the approach of sticking to the 10 classes 70,000 grayscale images in the size of $28 \times 28$ as in the original MNIST. In fact, the only change one needs to use this dataset is to change the URL from where the MNIST dataset is fetched. Moreover, Fashion-MNIST poses a more challenging classification task than the simple MNIST digits data, whereas the latter has been trained to accuracies above 99.7% as reported in [Wan et al., 2013, Ciregan et al., 2012].

We also looked at the EMNIST dataset provided by Cohen et al. [2017], an extended version of MNIST that extends the number of classes by introducing uppercase and lowercase characters. How-
ever, to be able to use it seamlessly one needs to not only extend the deep learning framework’s MNIST helpers, but also change the underlying deep neural network to classify these extra classes.

2 Fashion-MNIST Dataset

Fashion-MNIST is based on the assortment on Zalando’s website. Every fashion product on Zalando has a set of pictures shot by professional photographers, demonstrating different aspects of the product, i.e. front and back looks, details, looks with model and in an outfit. The original picture has a light-gray background (hexadecimal: #fdfdfd) and stored in 762 × 1000 JPEG format. For efficiently serving different frontend components, the original picture is resampled with multiple resolutions, e.g. large, medium, small, thumbnail and tiny.

We use the front look thumbnail images of 70,000 unique products to build Fashion-MNIST. Those products come from different gender groups: men, women, kids and neutral. In particular, white-color products are not included in the dataset as they have low contrast to the background. The thumbnails (51 × 73) are then fed into the following conversion pipeline, which is visualized in Figure 1.

1. Converting the input to a PNG image.
2. Trimming any edges that are close to the color of the corner pixels. The “closeness” is defined by the distance within 5% of the maximum possible intensity in RGB space.
3. Resizing the longest edge of the image to 28 by subsampling the pixels, i.e. some rows and columns are skipped over.
4. Sharpening pixels using a Gaussian operator of the radius and standard deviation of 1.0, with increasing effect near outlines.
5. Extending the shortest edge to 28 and put the image to the center of the canvas.
6. Negating the intensities of the image.
7. Converting the image to 8-bit grayscale pixels.

![Conversion Pipeline](image.png)

Figure 1: Diagram of the conversion process used to generate Fashion-MNIST dataset. Two examples from dress and sandals categories are depicted, respectively. Each column represents a step described in section 2.

Table 1: Files contained in the Fashion-MNIST dataset.

| Name                          | Description        | # Examples | Size     |
|-------------------------------|--------------------|------------|----------|
| train-images-idx3-ubyte.gz    | Training set images| 60,000     | 25 MBytes|
| train-labels-idx1-ubyte.gz     | Training set labels| 60,000     | 140 Bytes |
| t10k-images-idx3-ubyte.gz      | Test set images    | 10,000     | 4.2 MBytes|
| t10k-labels-idx1-ubyte.gz      | Test set labels    | 10,000     | 92 Bytes  |

For the class labels, we use the silhouette code of the product. The silhouette code is manually labeled by the in-house fashion experts and reviewed by a separate team at Zalando.

---

2 Zalando is the Europe’s largest online fashion platform. [http://www.zalando.com](http://www.zalando.com)
contains only one silhouette code. Table 2 gives a summary of all class labels in Fashion-MNIST with examples for each class.

Finally, the dataset is divided into a training and a test set. The training set receives a randomly-selected 6,000 examples from each class. Images and labels are stored in the same file format as the MNIST data set, which is designed for storing vectors and multidimensional matrices. The result files are listed in Table 1. We sort examples by their labels while storing, resulting in smaller label files after compression comparing to the MNIST. It is also easier to retrieve examples with a certain class label. The data shuffling job is therefore left to the algorithm developer.

Table 2: Class names and example images in Fashion-MNIST dataset.

| Label | Description     | Examples |
|-------|-----------------|----------|
| 0     | T-Shirt/Top     | ![Example Image](image1.png) |
| 1     | Trouser         | ![Example Image](image2.png) |
| 2     | Pullover        | ![Example Image](image3.png) |
| 3     | Dress           | ![Example Image](image4.png) |
| 4     | Coat            | ![Example Image](image5.png) |
| 5     | Sandals         | ![Example Image](image6.png) |
| 6     | Shirt           | ![Example Image](image7.png) |
| 7     | Sneaker         | ![Example Image](image8.png) |
| 8     | Bag             | ![Example Image](image9.png) |
| 9     | Ankle boots     | ![Example Image](image10.png) |

3 Experiments

We provide some classification results in Table 3 to form a benchmark on this data set. All algorithms are repeated 5 times by shuffling the training data and the average accuracy on the test set is reported. The benchmark on the MNIST dataset is also included for a side-by-side comparison. A more comprehensive table with explanations on the algorithms can be found on [https://github.com/zalandoresearch/fashion-mnist](https://github.com/zalandoresearch/fashion-mnist).

Table 3: Benchmark on Fashion-MNIST (Fashion) and MNIST.

| Classifier         | Parameter                                                                 | Fashion | MNIST |
|--------------------|---------------------------------------------------------------------------|---------|-------|
| DecisionTreeClassifier | criterion=entropy max_depth=10 splitter=best | 0.798   | 0.873 |
|                     | criterion=entropy max_depth=10 splitter=random | 0.792   | 0.861 |
|                     | criterion=entropy max_depth=50 splitter=best | 0.789   | 0.886 |

Continued on next page
Table 3 – continued from previous page

| Classifier          | Parameter                   | Test Accuracy |  |
|---------------------|-----------------------------|---------------|---|
|                     |                             | Fashion       | MNIST |
|                     | criterion=entropy max_depth=10 splitter=best | 0.789         | 0.886 |
|                     | criterion=entropy max_depth=10 splitter=best | 0.788         | 0.866 |
|                     | criterion=entropy max_depth=50 splitter=best | 0.787         | 0.883 |
|                     | criterion=entropy max_depth=100 splitter=best | 0.787         | 0.881 |
|                     | criterion=entropy max_depth=10 splitter=best | 0.787         | 0.879 |
|                     | criterion=entropy max_depth=100 splitter=best | 0.783         | 0.877 |
|                     | criterion=entropy max_depth=50 splitter=best | 0.783         | 0.853 |
|                     | criterion=entropy max_depth=10 splitter=best | 0.779         | 0.873 |
|                     | criterion=entropy max_depth=100 splitter=best | 0.777         | 0.875 |
| **ExtraTreeClassifier** | criterion=gini max_depth=10 splitter=best | 0.775         | 0.806 |
|                     | criterion=entropy max_depth=10 splitter=best | 0.775         | 0.847 |
|                     | criterion=entropy max_depth=10 splitter=best | 0.772         | 0.810 |
|                     | criterion=entropy max_depth=50 splitter=best | 0.772         | 0.847 |
|                     | criterion=gini max_depth=10 splitter=best | 0.769         | 0.843 |
|                     | criterion=gini max_depth=50 splitter=best | 0.768         | 0.845 |
|                     | criterion=gini max_depth=50 splitter=best | 0.752         | 0.826 |
|                     | criterion=gini max_depth=100 splitter=best | 0.752         | 0.828 |
|                     | criterion=gini max_depth=50 splitter=best | 0.748         | 0.824 |
|                     | criterion=gini max_depth=100 splitter=best | 0.745         | 0.820 |
|                     | criterion=gini max_depth=10 splitter=best | 0.739         | 0.737 |
|                     | criterion=gini max_depth=100 splitter=best | 0.737         | 0.745 |
| **GaussianNB**      | priors=[0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1] | 0.511         | 0.524 |
| **GradientBoostingClassifier** | n_estimators=100 loss=deviance max_depth=10 | 0.880         | 0.969 |
|                     | n_estimators=50 loss=deviance max_depth=10 | 0.872         | 0.964 |
|                     | n_estimators=100 loss=deviance max_depth=3 | 0.862         | 0.949 |
|                     | n_estimators=10 loss=deviance max_depth=10 | 0.849         | 0.933 |
|                     | n_estimators=50 loss=deviance max_depth=3 | 0.840         | 0.926 |
|                     | n_estimators=10 loss=deviance max_depth=50 | 0.795         | 0.888 |
|                     | n_estimators=10 loss=deviance max_depth=3 | 0.782         | 0.846 |
| **KNeighborsClassifier** | weights=distance n_neighbors=5 p=1 | 0.854         | 0.959 |
|                     | weights=distance n_neighbors=9 p=1 | 0.854         | 0.955 |
|                     | weights=uniform n_neighbors=9 p=1 | 0.853         | 0.955 |
|                     | weights=uniform n_neighbors=5 p=1 | 0.852         | 0.957 |
|                     | weights=distance n_neighbors=5 p=2 | 0.852         | 0.945 |
|                     | weights=distance n_neighbors=9 p=2 | 0.849         | 0.944 |
|                     | weights=uniform n_neighbors=9 p=2 | 0.849         | 0.944 |
|                     | weights=distance n_neighbors=1 p=2 | 0.847         | 0.943 |
|                     | weights=distance n_neighbors=1 p=1 | 0.839         | 0.943 |
|                     | weights=uniform n_neighbors=1 p=1 | 0.838         | 0.955 |
|                     | weights=distance n_neighbors=1 p=1 | 0.838         | 0.955 |
| **LinearSVC**       | loss=hinge C=1 multi_class=ovr penalty=12 | 0.836         | 0.917 |
|                     | loss=hinge C=1 multi_class=crammer_singer penalty=12 | 0.835         | 0.919 |
|                     | loss=squared_hinge C=1 multi_class=crammer_singer penalty=12 | 0.834         | 0.919 |
|                     | loss=squared_hinge C=1 multi_class=crammer_singer penalty=11 | 0.833         | 0.919 |
|                     | loss=squared_hinge C=1 multi_class=crammer_singer penalty=11 | 0.833         | 0.919 |
|                     | loss=squared_hinge C=1 multi_class=ovr penalty=12 | 0.820         | 0.912 |
|                     | loss=squared_hinge C=10 multi_class=ovr penalty=12 | 0.779         | 0.885 |
|                     | loss=squared_hinge C=100 multi_class=ovr penalty=12 | 0.776         | 0.873 |
|                     | loss=hinge C=10 multi_class=ovr penalty=12 | 0.764         | 0.879 |
|                     | loss=hinge C=100 multi_class=ovr penalty=12 | 0.758         | 0.872 |

Continued on next page
| Classifier             | Parameter                                      | Test Accuracy |
|------------------------|-----------------------------------------------|---------------|
|                        |                                               | Fashion       | MNIST        |
|                        |                                               | 0.751         | 0.783        |
|                        |                                               | 0.749         | 0.816        |
|                        |                                               | 0.748         | 0.829        |
|                        |                                               | 0.736         | 0.829        |
|                        |                                               | 0.516         | 0.759        |
|                        |                                               | 0.496         | 0.753        |
|                        |                                               | 0.492         | 0.746        |
|                        |                                               | 0.484         | 0.737        |
| LogisticRegression     | C=1 multi_class=ovr penalty=11                | 0.842         | 0.917        |
|                        | C=1 multi_class=ovr penalty=12                | 0.841         | 0.917        |
|                        | C=10 multi_class=ovr penalty=12               | 0.839         | 0.916        |
|                        | C=10 multi_class=ovr penalty=11               | 0.839         | 0.909        |
|                        | C=100 multi_class=ovr penalty=12              | 0.836         | 0.916        |
| MLPClassifier          | activation=relu hidden_layer_sizes=[100]      | 0.871         | 0.972        |
|                        | activation=relu hidden_layer_sizes=[100, 10]  | 0.870         | 0.972        |
|                        | activation=tanh hidden_layer_sizes=[100]      | 0.868         | 0.962        |
|                        | activation=tanh hidden_layer_sizes=[100, 10]  | 0.863         | 0.957        |
|                        | activation=relu hidden_layer_sizes=[10]       | 0.850         | 0.936        |
|                        | activation=relu hidden_layer_sizes=[10, 10]   | 0.848         | 0.933        |
|                        | activation=tanh hidden_layer_sizes=[10]       | 0.841         | 0.921        |
|                        | activation=tanh hidden_layer_sizes=[10, 10]   | 0.840         | 0.921        |
| PassiveAggressiveClassifier | C=1                                           | 0.776         | 0.877        |
|                        | C=100                                          | 0.775         | 0.875        |
|                        | C=10                                           | 0.773         | 0.880        |
| Perceptron             | penalty=1                                      | 0.782         | 0.887        |
|                        | penalty=12                                     | 0.754         | 0.845        |
|                        | penalty=elasticnet                             | 0.726         | 0.845        |
| RandomForestClassifier | n_estimators=100 criterion=entropy max_depth=100 | 0.873         | 0.970        |
|                        | n_estimators=100 criterion=entropy max_depth=100 | 0.872         | 0.970        |
|                        | n_estimators=50 criterion=entropy max_depth=100 | 0.872         | 0.968        |
|                        | n_estimators=100 criterion=entropy max_depth=50 | 0.872         | 0.969        |
|                        | n_estimators=50 criterion=entropy max_depth=50 | 0.871         | 0.967        |
|                        | n_estimators=100 criterion=gini max_depth=100  | 0.871         | 0.971        |
|                        | n_estimators=50 criterion=gini max_depth=100  | 0.870         | 0.968        |
|                        | n_estimators=50 criterion=gini max_depth=100  | 0.869         | 0.967        |
|                        | n_estimators=10 criterion=entropy max_depth=50 | 0.853         | 0.949        |
|                        | n_estimators=10 criterion=entropy max_depth=100 | 0.852         | 0.949        |
|                        | n_estimators=10 criterion=gini max_depth=100  | 0.848         | 0.948        |
|                        | n_estimators=10 criterion=gini max_depth=100  | 0.847         | 0.948        |
|                        | n_estimators=50 criterion=entropy max_depth=100 | 0.838         | 0.947        |
|                        | n_estimators=100 criterion=entropy max_depth=100 | 0.838         | 0.950        |
|                        | n_estimators=100 criterion=gini max_depth=100  | 0.835         | 0.949        |
|                        | n_estimators=50 criterion=gini max_depth=100  | 0.834         | 0.945        |
|                        | n_estimators=10 criterion=entropy max_depth=100 | 0.828         | 0.933        |
|                        | n_estimators=10 criterion=gini max_depth=100  | 0.825         | 0.930        |
| SGDClassifier          | loss=hinge penalty=12                         | 0.819         | 0.914        |
|                        | loss=perceptron penalty=11                   | 0.818         | 0.912        |
|                        | loss=modified_huber penalty=11               | 0.817         | 0.910        |
|                        | loss=modified_huber penalty=12               | 0.816         | 0.913        |
|                        | loss=log penalty=elasticnet                  | 0.816         | 0.912        |
|                        | loss=hinge penalty=elasticnet                | 0.816         | 0.913        |

Continued on next page
Table 3 – continued from previous page

| Classifier | Parameter | Test Accuracy |
|------------|-----------|---------------|
|            |           | Fashion | MNIST |
| loss=squared_hinge penalty=elasticnet | 0.815 | 0.914 |
| loss=hinge penalty=11 | 0.815 | 0.911 |
| loss=log penalty=11 | 0.815 | 0.910 |
| loss=perceptron penalty=12 | 0.814 | 0.913 |
| loss=perceptron penalty=elasticnet | 0.814 | 0.912 |
| loss=squared_hinge penalty=12 | 0.814 | 0.912 |
| loss=modified_huber penalty=elasticnet | 0.813 | 0.914 |
| loss=log penalty=12 | 0.813 | 0.913 |
| loss=squared_hinge penalty=11 | 0.813 | 0.911 |
| SVC | C=10 kernel=rbf | 0.897 | 0.973 |
|         | C=10 kernel=poly | 0.891 | 0.976 |
|         | C=100 kernel=poly | 0.890 | 0.978 |
|         | C=100 kernel=rbf | 0.890 | 0.972 |
|         | C=1 kernel=rbf | 0.879 | 0.966 |
|         | C=1 kernel=poly | 0.873 | 0.957 |
|         | C=1 kernel=linear | 0.839 | 0.929 |
|         | C=10 kernel=linear | 0.829 | 0.927 |
|         | C=100 kernel=linear | 0.827 | 0.926 |
|         | C=1 kernel=sigmoid | 0.678 | 0.898 |
|         | C=10 kernel=sigmoid | 0.671 | 0.873 |
|         | C=100 kernel=sigmoid | 0.664 | 0.868 |

4 Conclusions

This paper introduced Fashion-MNIST, a fashion product images dataset intended to be a drop-in replacement of MNIST and whilst providing a more challenging alternative for benchmarking machine learning algorithm. The images in Fashion-MNIST are converted to a format that matches that of the MNIST dataset, making it immediately compatible with any machine learning package capable of working with the original MNIST dataset.

References

D. Ciregan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 3642–3649. IEEE, 2012.

G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. Emnist: an extension of mnist to handwritten letters. arXiv preprint arXiv:1702.05373, 2017.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 248–255. IEEE, 2009.

A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus. Regularization of neural networks using dropconnect. In Proceedings of the 30th international conference on machine learning (ICML-13), pages 1058–1066, 2013.