MATH4432: Final Project

Transfer Learning on Fashion-MNIST

Alexander Moellers 20501050

Nora Eliza Norden 20500343

Tuesday 22nd May, 2018

1 Introduction

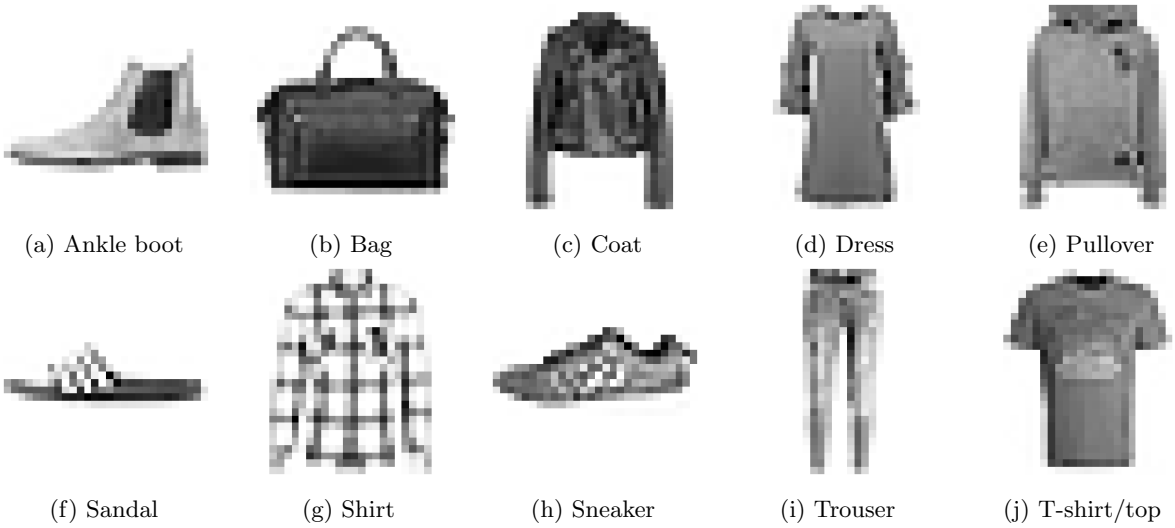
The main goal of this project is to implement a machine learning procedure that can successfully classify images of clothes from the Fashion-MNIST dataset [1]. For this, a neural network is used to extract different features from the dataset and then a classification algorithm is trained to make a prediction based on these features.

The approach used is called Transfer learning which describes the idea of using the knowledge gained with one method and apply it to help the next learner achieve better results. In this case the first approach is the neural network which extracts the features of the images so that the classification algorithm can use them. For this project, the PyTorch neural network ResNet-18 is used for feature extraction. The raw image data provides an alternative set of features so that the performances can be compared. At this point it is to be mentioned that a ResNet-18 is a pretrained model and thus can also be considered a form of transfer learning.

The code was written in Python notebooks using Google Colaboratory to access GPU. This comes with the advantage that several parties can simultaneously work on the project and that the computations are performed on the Google servers. All code is available on Github.

2 Dataset

The Fashion-MNIST dataset is becoming more and more popular as the standard MNIST dataset becomes too easy for state-of-the-art machine learning algorithms. The dataset contains images of 70,000 fashion items. Out of these, 60,000 images are training data and the remaining belong to the test dataset. Each grayscale image is 28 by 28 pixels. There are a total of 10 disjoint classes that each item can belong to. These are: ankle boot, bag, coat, dress, pullover, sandal, shirt, sneaker, trouser and t-shirt/top. Figure 1 shows how items of each class may look like.



(a) Ankle boot (b) Bag (c) Coat (d) Dress (e) Pullover

(f) Sandal (g) Shirt (h) Sneaker (i) Trouser (j) T-shirt/top

Figure 1: Sample items from each of the ten classes.

3 Neural Network for Feature Extraction

The reason why a neural network is used for feature extraction is because it enables the computer to learn from the input data. That is, it can start to recognize patterns in, for example, images. To accomplish this task the neural network makes use of its different layers, where each layer is made up of several nodes, see fig. 2. The first layer is the so called input layer and the last layer is the output layer. In the case of the Fashion-MNIST data set one can imagine the input nodes as the different pixels. Each node is assigned a value which represents the brightness of the pixel. The output layer nodes are the ten classes such as sandal or coat. In between these two layers, hidden layers exit, however they do not have a specific meaning. One can imagine that the algorithm extracts features in these layers, which might be angles or other geometrical structures. Between the nodes of different layers connections exists and during the training process these are weighted in a way that the activation of pixels for a given input image in the first layer leads to the activation of the node that represent the class of that image in the output layer [2].

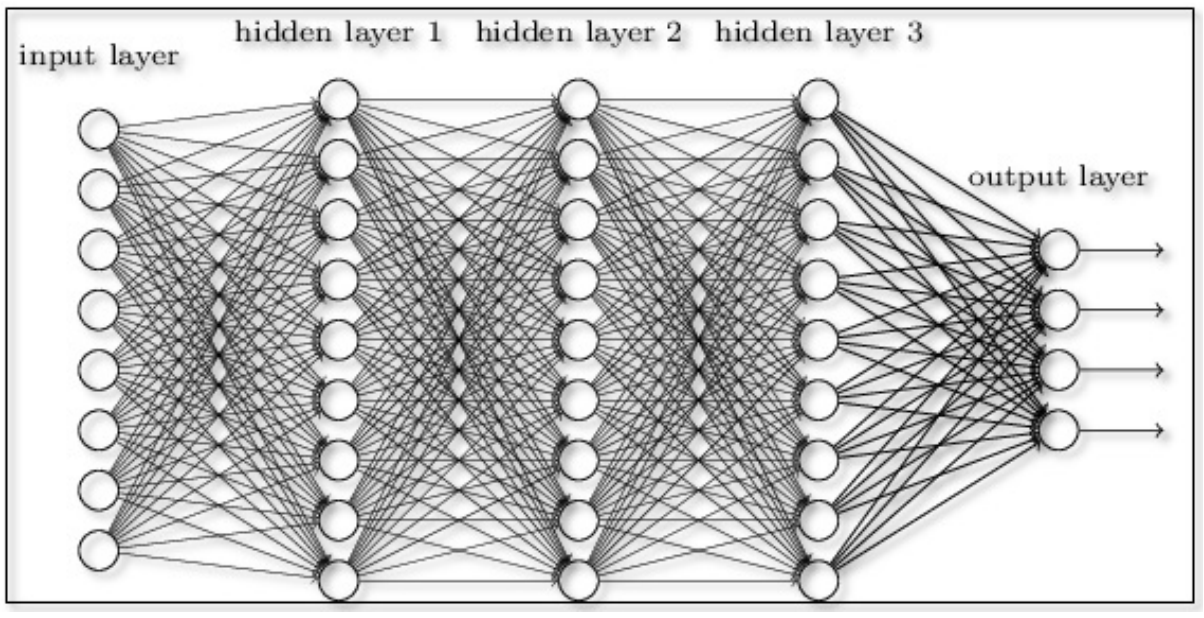


Figure 2: Neural net with three hidden layers. Image credit: Mathworks.

If one now disregards the output layer and takes the nodes of the penultimate layer than these can be considered features of the images. Depending on the activation of these nodes or features can be used to predict which class an image belongs to. A classification algorithm such as Random Forrest or Support Vector Machines can then be used to predict the class of a non-labeled image based on these features.

The neural network used for this project is ResNet-18. ResNet-18 is a deep residual learning model in the PyTorch library, pre-trained on the ImageNet dataset to classify images amongst 22,000 hierarchical classes [3]. Some of the most popular classes in ImageNet include animals, plants, foods and sports. However, it does not contain any fashion item classes, and therefore, ResNet-18 has not been trained to classify these. The model achieves 3.57 % test error on the ImageNet dataset, which yielded a winning position in the Large Scale Visual Recognition Challenge (ILSVRC) 2015 [4].

ResNet-18 is the first model to utilize the deep residual learning framework, and the approach was first introduced by Microsoft researchers Kaiming He, Xiangyu Zhang Shaoqing Ren and Jian Sun in 2015 [4]. One of the most crucial hyper parameters in neural networks is the depth, or number of layers, of the model. Deeper networks imply that the program can solve more complex tasks, but deep networks may also suffer from the so called degradation problem. In short, large depth leads to a saturated accuracy, which happens when the model stops improving. However, further increasing depth leads to a rapid degradation in accuracy that is not caused by overfitting [5].

The objective of a neural network, or any other machine learning model, is to fit to the true underlying mapping of the data. In a traditional neural network, with layers directly stacked upon each other, each layer is trying to optimize a zero mapping $F(x) = 0$ to the data. A residual mapping, on the other hand, is defined as $H(x) = F(x) + I(x)$, where $F(x)$ is the mapping achieved by the earlier models and $I(x) = x$ an identity mapping. A residual network adds these residual mappings, see fig. 3, and tries to optimize $H(x)$ instead of $F(x)$. The mapping is based on the assumption that it is easier to find a model closer to an identity mapping than to a zero mapping, and hence, the optimization is simplified. Deep residual networks are also guaranteed to be protected against the degradation problem, since a layer may at worst find the function $F(x)$, the mapping from the shallower network.

Using a pretrained model can save a lot of time, because the weights of the connections between nodes do not have to be initalized at random. Instead they are initialized with values from a similar dataset and then adjusted to fit the input data. The model obtained by training a neural network on the ImageNet dataset can thus be used to greatly facilitate the feature extraction from the Fashion-MNIST dataset.

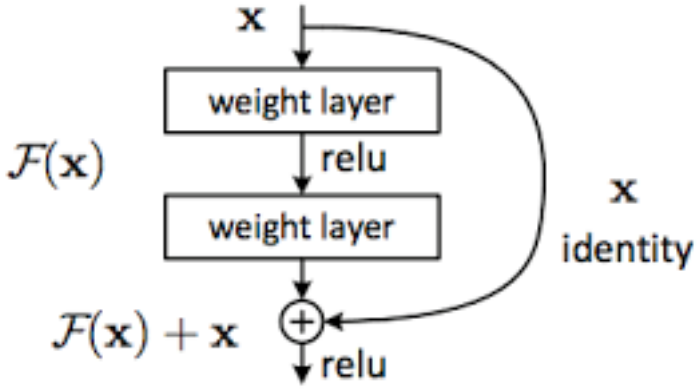


Figure 3: A simple residual network with a residual mapping over two layers. Image credits: He, Zhang, Ren and Sun, https://arxiv.org/pdf/1512.03385.pdf.

4 Visualization

The feature space of an image is $28 \times 28 = 784$ dimensions, and the ResNet model extracts 512 unique features. A simple way to visualize the different classes is to plot the data points in two dimensions after applying PCA, or principal component analysis. PCA finds the vectors along which the data variance is

Table 1: Color map of the different classes.

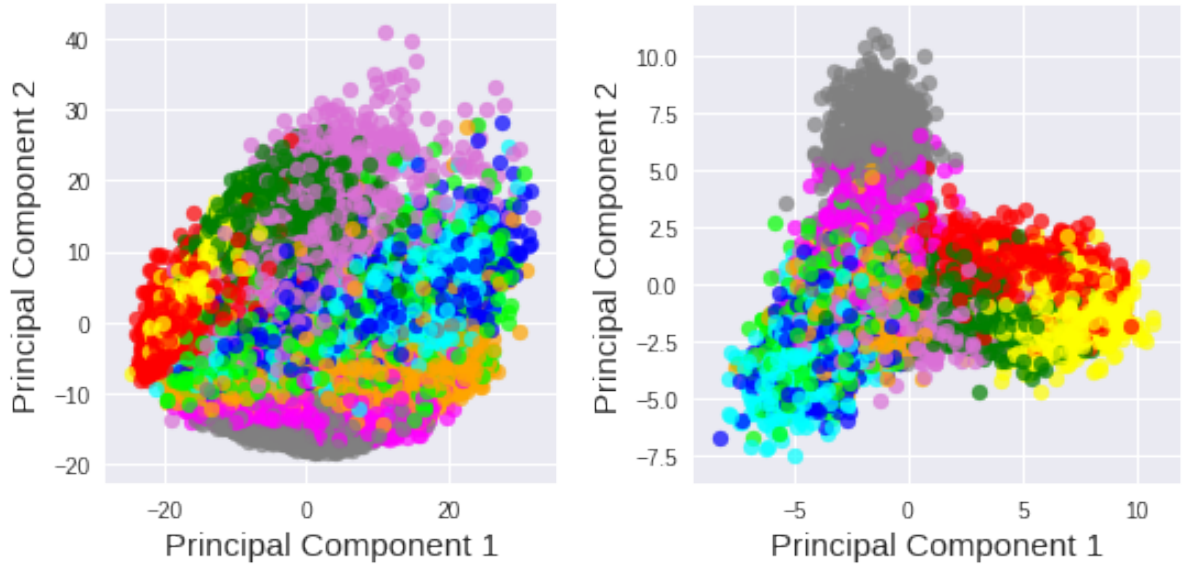
Class	Color
Ankle Boot	Green
Bag	Orchid
Coat	Cyan
Dress	Magenta
Pullover	Blue
Sandal	Red
Shirt	Lime
Sneaker	Yellow
Trouser	Gray
T-shirt/top	Orange

maximized, thus capturing the most data differences in a reduced number of dimensions. Before the data is transformed, it is standardized by subtracting the mean and scaling to unit variance.

Each class is mapped to a color according to table 1. The result of the PCA can be seen in fig. 4. Although the raw data-PCA exhibits some inter-class differences, the data points are much more clustered than for the ResNet features, making it more difficult to distinguish the classes.

The scatter plot of the two principal components of the ResNet features, seen in fig. 4b, neatly separates the classes into three major directions. On top, only trousers are found.

Shoes (ankle boots, sandals and sneakers) are found to the right. The leftmost points are mainly coats, pullovers and shirts. In the middle, we find dresses, bags, and t-shirts/tops. Note that the raw image data PCA does a similar separation, albeit the class boundaries are not as clearly defined.



(a) Raw data

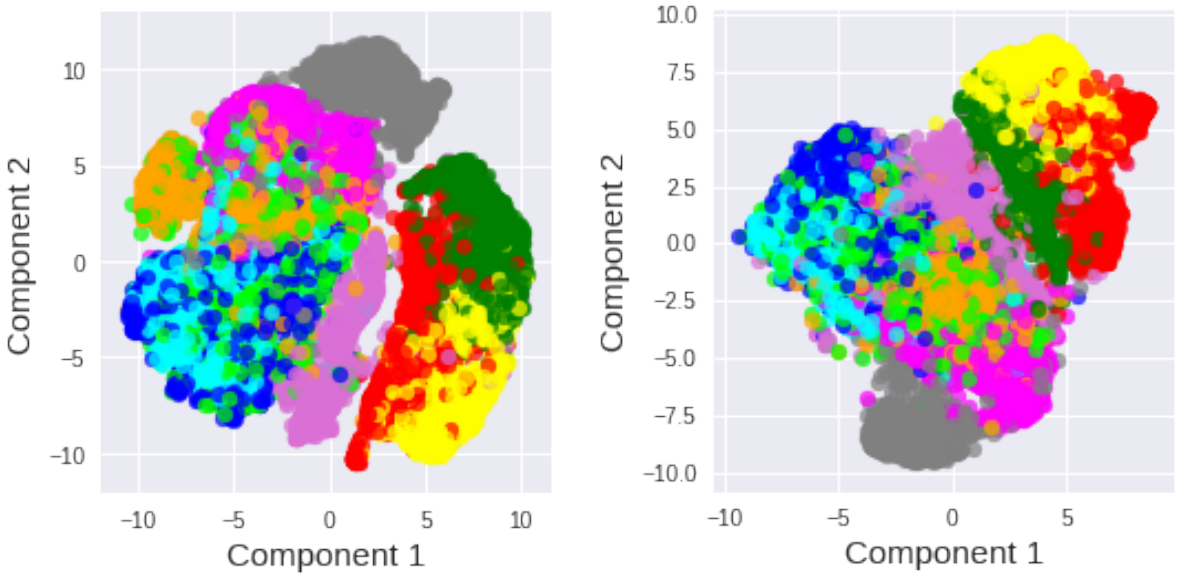
(b) ResNet

Figure 4: PCA performed on the raw image data and the ResNet-extracted features.

Another way to visualize the data is to use t-distributed stochastic neighbour embedding (t-SNE). This is a technique for dimensional reduction which can be used for datasets with multiple variables. It attempts to preserve the clustering and maintain as much information as possible. The approach is computing the similarity between points by making use of the distance and mapping it under a normal distribution. Points that are closer to the starting point will have higher similarity values. Points with high similarity values can then be drawn close to each other in lower dimensional spaces. The similarity of the points in the lower dimensional space will be computed using a t-distribution and the points will be moved to fit the similarity determined before in the higher dimension. This will result in a figure in which clustered point are drawn close to each other.

Before implementing the t-SNE algorithm the dimension were reduced to 50 using principal component analysis. This is recommended by the creators of t-SNE because the algorithm is less effective when more than 50 dimensions are used as input. The embeddings can be seen in fig. 5. Here several classes are especially well clustered, these are Trousers, Ankle boots, Sandals and Sneakers. Coats and Pullovers seem to be mixed up and this might hint at a difficult situation for the algorithm that has to distinguish between the two.

Generally, fig. 5 shows that data points from the same class are successfully grouped together. Clusters can be distinguished and this gives an indication that the features used have predictive value. The visualization achieved with T-SNE seems to depict the clusters clearer than the principal component analysis alone. Furthermore, the clusters found by using the features from the raw data appear as least as good and maybe better than the ones found based on the ResNet features.



(a) Raw data t-SNE Visualization

(b) ResNet features t-SNE visualization

Figure 5: t-SNE performed on the raw image data and the ResNet-extracted features.

5 Classification results

The classification models used to classify the ResNet features and the raw image data are; LDA, QDA, logistic regression, linear SVC and random forest. Some of these take hyper parameters as argument, and those have been determined by grid search cross validation. Due to computation time and limited GPU memory on the virtual machine, the grid is limited to 2-3 values per parameter. Therefore, the tuning of hyper parameters is fairly rough. The accuracies yielded by each classification model are summarized in table 2.

The linear and quadratic discriminant analysis do not require any hyper parameter tuning. As seen in table 2, LDA proved to be better than QDA. The low accuracy of the QDA classification on the raw data

Table 2: Accuracies of the different classification models using the best hyper parameters.

Model	ResNet	Raw data
LDA	0.7708	0.8151
QDA	0.7661	0.5680
Logistic regression	0.7771	0.8016
Linear SVC	0.7702	0.7687
Random forest	0.6263	0.8763

can be explained by variables being colinear, which raised a warning in Python. For the ResNet features, LDA is marginally better than QDA. QDA is the more flexible model of the two, and the different results can potentially be explained by the regularizing property of LDA. However, the difference is too small to know with certainty. We also note that the LDA model performs better on the raw data than the extracted features.

The tested logistic regression hyper parameters were the regularization strength, C, and the maximum number of iterations, max_iter. C can be either 0.1, 1.0 or 5.0, the maximum number of iterations 100 or 200. Grid search CV for the both models showed that the best combination were $C = 1.0$ and max_iter $= 100$.

Grid search cross validation for the linear support vector classifiers implied that highest accuracy was achieved for $C = 1.0$ and max_iter $= 1000$. In this grid search, C could take the values 0.1 and 1.0, and max_iter the values 500 and 1000.

Both random forests models chose the parameters no max depth and the max features used equaling the square root of the total features. However, the raw data model achieved a remarkable 0.25 points more in accuracy.

6 Conclusion

It turned out that the ResNet classification results were worse or just marginally better than classifying the raw image data. It implies that the ResNet features are not very well suited for the Fashion-MNIST dataset, and the last layers would require some retraining to achieve a performance that can compete with the raw image classification. This would indeed be an interesting task that wasn't feasible for this project, due to computational time and limited access to GPU.

As mentioned earlier, another key factor for the accuracy was the hyperparameter choice. Grid search cross validation always gave the same parameters as the Python Scikit library's default parameter values. A finer grid and a larger number of hyperparameters cross validated could potentially find parameter values that boosted the model performances.

To conclude, ResNet-18 extracted features yielded a performance drop as compared to the raw data classifications. Undoubtedly, a neural network alone could achieve higher accuracy and so it seems that using a different classification algorithm on the penultimate layer might not yield the best possible performance. Nevertheless, an advantage of transfer learning became clear - it can dramatically speed up

training and computation times. With more time, higher accuracy should also be achievable, but more research would have to be conducted in fine tuning the different models.

7 Individual contributions

The workload was evenly split between the group members.

References

[1] Han Xiao, Kashif Rasul and Roland Vollgraf, *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*, https://github.com/zalandoresearch/fashion-mnist, 2017-08-28.

[2] Ankit Gupta, *What is Deep Learning and Neural Networks?*, http://www.thewindowsclub.com retrieved 2018-05-20.

[3] *ImageNet*, http://image-net.org, retrieved 2018-05-19.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, *Deep Residual Learning from Image Recognition*, 2015-12-10.

[5] Michael Dietz, *Understand Deep Residual Networks - a simple, modular learning framework that has redefined state-of-the-art*, https://blog.waya.ai/deep-residual-learning-9610bb62c355, 2017-05-02.