Assignment 2: Machine Learning project: Fashion Mnist

S3629363 Jiewen Guan

Summary

This project has explored the performance of MLP, CNN, decision tree and random forest classifier based on the fashion MNIST dataset. After tuning varies parameter and structures, the test accuracy has reached 89% for MLP, 94% for CNN, 81% for decision tree and 88% for random forest.

I. Design

a. The dataset

The dataset was developed by the Zalando company, as a drop-in replacement for the popular MNIST handwriting dataset. As the original MNIST handwriting, the fashion-MNIST has 70000 instances of 28*28 images scattered over 10 categories. However, the idea of making it interchangeable with the MNIST- handwriting, which must have the exact same structure, may cause problem.

First, the category selection of this dataset is not well considered, as the distinguish between t shirt, sweater(pullover), and coats are not clear as the distinguish between handwritten numbers. Besides that, the compression of the image could also lead to problems. For the original MNIST-handwriting dataset, a 28*28 image is a good enough representation of the handwritten digits, as they are 2 dimensional shapes formed by lines. However, for the MNIST fashion, these 70000 of highly compressed images are images of common clothing, which are 3d objects by nature, and may have varies different colours. Due to the compression, a good portion of information is lost, some images in different categories are visually similar.

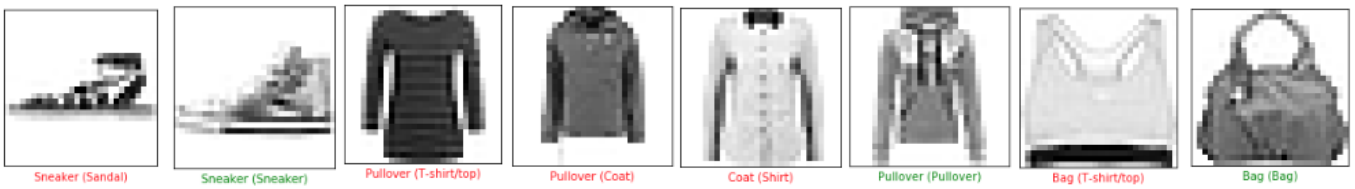


Fig 1. Visual similarities

The other issue of this dataset is that all the items are in frount-centered position and all the images have a complete 0 value as background, therefore, using the models trained on this dataset to classify realworld clothings may lead to poor accuracy, or else the image of the object to be classified must meet varies standards of the database.

As shown in the Benchmark section of the github page of the dataset (1), human performance on this dataset is around 0.835, which means there is a concidrable amount of noise embedded in this datset that even the most sofisticated neuronetwork on this planet can not distinguish.

Aside of the disadvantages above, this dataset also have many good charesteristics that may help the training. For example, it has no missing values, no temproal ralations, 6000 training samples per category. Which means it could fit to many supervized machine learning models out of the box without much of preproccesing and enviroment construction.

In conclusion, due to the reasons above, this toy dataset is not a good enough representation of realworld common clothings, but it could be a good toy model for the starters, the model trained on it should only be used for classifications on the subset of the dataset itself.

b. Design considerations

Due to it being a popular dataset, many people have attempted a lot of models on it, and quite a few of them is listed on the benchmark page [2] of the model. Among all of them, the WRN40-4 8.9M params has the best test accuracy of 0.967, however, the WRN40 is a wide residual network with 40 layers and unconventional connections between the layers. The other normal CNNs particularly has an accuracy around 0.9-0.94, in some case, batch

normalization and dropouts may have contributed 0.01-0.02 of the accuracy. Other than these, the standard fully connected MLP have a best accuracy of 0.88.

Besides all the neural-network-based models, the best performance of 0.897 is produced by the SVC, which is a multi-class version of support vector machine. However, due to the number of data instances in this dataset, and the vector calculations, it takes over 1 hour to train the model. The second place goes to the gradient boosting classifier. However, it has whooping 17 hours of training time, the reason behind the long training time might be it is recursively building 100 decision trees, each trained on the mistake of the previous one.

The third best performance is from a random forest classifier, which also has 100 decision trees, however, due to the number of features that one tree concerns is only a subset of the entire image, and there is no dependencies between trees, the trees could be trained in parallelly and the training time is reduced to 8 minutes. It is reasonable to use the random forest on this kind of image recognition tasks, due to its similarity with the CNN networks. The difference between these two models is that a convolutional filter takes in features from adjacent pixels, but the trees would take in purely random subset of pixels across the whole image. As the convolutional filters may be good at recognize small features in small adjacent areas, in theory the trees could also recognize features that scattered around the entire image if it happened to select the right subset of pixels to train. The difference of these two models would be that in the end the random forest will sum up the opinions of all the trees by voting without any calculation of weights and certainly has no fully connected layers to adjust the weights by backward propagation. There is attempts [3] to use random forest to construct feature for a deep network, but in this paper, the two systems are separated as the deep network don't affect the random forest by any means. Besides that, before the age of neural nets, there was well developed computer vision products such as the Xbox Kinect [4] that utilize the random forest algorithm to perform body posture recognition.

c. Model selection

Due to the limitation of the timeframe, processing power and the dataset itself. I decided to explore the following models.

i. CNN

As the CNN is the current go to method for image recognition, there is no reason not to try this model. However, the scope of CNN in this project is limited to neural networks that has no more than around 10 layers that consist of convolution layers and fully connected dense layers. Pooling, batch normalization and dropout will be applied if necessary. The other deep networks with special structures such as the ResNet, DenseNet will not be explored.

ii. MLP

The MLP could be a good reference point for the CNN model, as it could demonstrate the contribution of convolution layers. Batch normalization and dropout will be applied if needed.

iii. Random forest

As listed in the benchmark [2] the random forest is the best performing none-neural-network based model that also has a feasible training time, and it also has a similarity to the CNN model, it will be explored in this project.

iv. Decision tree

Regular single decision tree could be a good reference point for the random forest to demonstrate the contribution of the bagging strategy and multiple trees.

II. Investigation

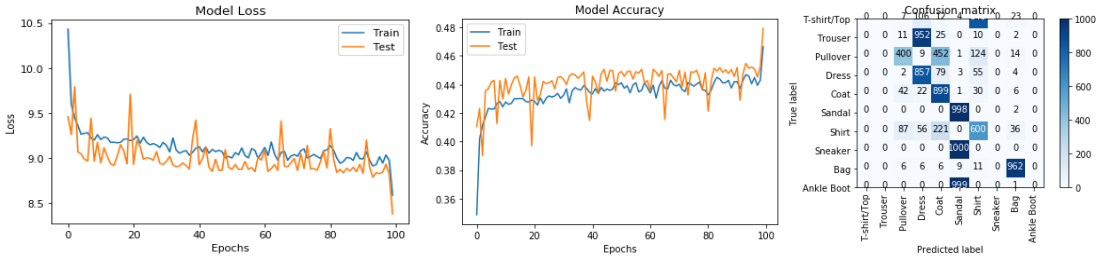
a. loading & Pre-processing

I decide to use the inbuilt loading functions in the keras library because it saves the struggle of managing local datafiles and directories, and it will always run when the internet is available. The labels are then one hot encoded for the neural nets but not for the decision trees. As doing so would massively unbalance the tree models. The image array is reshaped to 1d for the MLP and the trees as they can't take 2d array.

b. Training parameters & impact

i. MLP

There are several training parameters for the MLP model to be tuned, such as normalization, number of hidden nodes, number of hidden layers, dropout rate, optimizer, learning rate for optimizer and batch normalization. The most significant impact is brought by normalizing the data by dividing the feature array by 255. As shown in the 8th experiment, without the initial normalization, in the training the model starts at around 40% of accuracy and barely reached 50% of accuracy after 100 epoch and funny enough 100% of the sneakers and 99.9% of ankle boots were classified as sandals. Besides that, these are the major parameters.



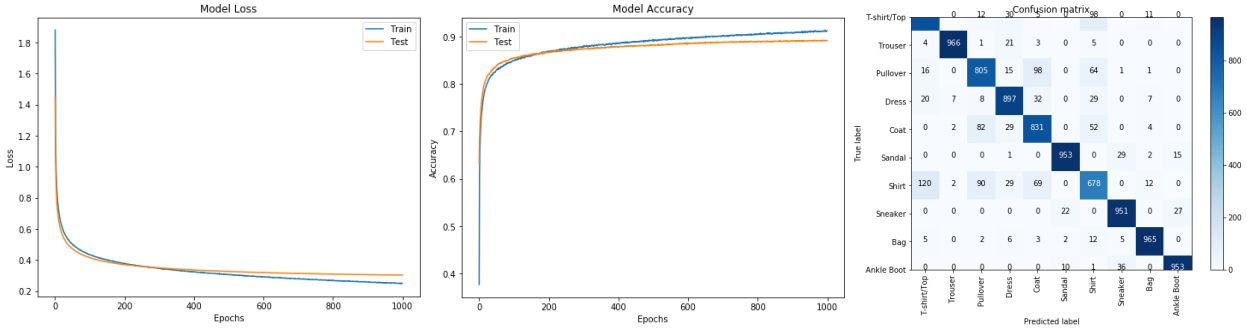
1. Layer width. as shown in the A2-MLP notebook, once passed 100 nodes, adding more nodes to a single layer did not produce major contribution to the validation accuracy, therefore all the model afterwards will only have a layer width of 100.

2. layer number. The second experiment in A2-MLP also demonstrate that more than one hidden 100 node layer did not bring noticeable positive impact to the test result. Therefore, all the model afterwards will only have 1 hidden layer.

3. dropout rate. Due to the overfitting observed in previous experiments, a dropout layer is added between the hidden layer and the output layer. The drop out rate of 0.1-0.7 is tested in 7 trains each has 0.1 higher dropout rate than the previous one. As shown in the notebook, this lobotomy decreased the training accuracy step by step, but the validation accuracy stayed in the same trend. however, the test accuracy is most optimal at the rate of 0.3.

4. batch size. In experiment 4, Smaller batch size have major negative impact on training speed, but only minor positive impact on test accuracy. Some say this is the reason many people can't reproduce the experiment result of Alex Graves because he always uses 1 for batch size.

5. optimizers. In experiment 5, 7 type of optimizer is tested with their default settings, it turns out that within 10 epochs, the Adam and Nadam optimizer has the best test accuracy. However, when plotting the training/validating performance, the stochastic gradient descent has the most beautiful continuous curve and no sign of overfitting, but with an 8% lower accuracy comparing to other models. In the experiment 5.5, a long-term test of SGD optimizer is conducted, it demonstrates that over 1000 epochs, the model with SGD optimizer could eventually reach an accuracy of around 89%, which is also the peak performance reached by models with Adam optimizer within 30 epochs. Therefore, the SDG optimizer is only affecting the training speed, but not the final result

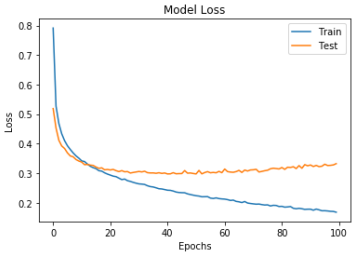


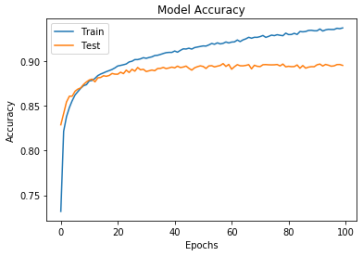
6. learning rate. The learning rate plays a major role in the training, in the experiment 6, 7 different learning rates are tested, it turns out that 0.0003 or 3e-4 is the best learning rate for models with Adam optimizer.

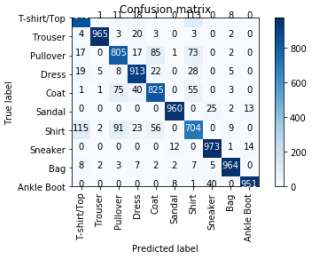
7. Batch normalization. A batch normalization layer is added between the hidden layer and the dropout layer.

As shown in the confusion matrix, it has increased accuracy for classifying shirts but decreased it for all others, the overall accuracy was dropped by 1%.

Final model. Based on all the experiments above, the final model consists of 1 hidden layer of 100 nodes, followed by a dropout layer of 0.3, then the output layer, it is compiled with Adam optimizer and a learning rate of 3e-4. It reaches its best validation performance around 50 epochs and the outcome has an accuracy of 89% and a loss of 0.34.





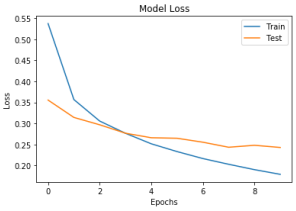


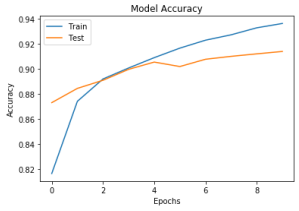
Conclusion for MLP. As shown by the confusion matrix and the classification report of the final model, the category with best performances is the trousers, bag and sandals, due to these categories has no visually similar category in the dataset, and the shirt, coat pullover and T-shirt do have miss classifications between them as proposed in the design section of this report.

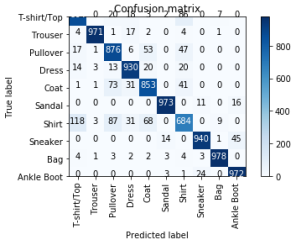
ii. CNN

As the fully connected layers of a CNN model are very similar to an MLP network, there is no point to re-explore them again, therefore I will use the best performing MLP model as the fully connected layer for the CNN model.

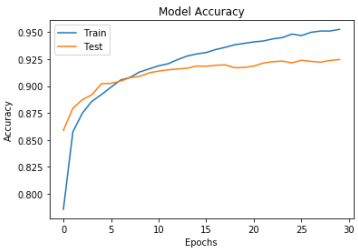
1. Convolution layer. As shown in the A2-CNN notebook, the first model is the best MLP model with one 3*3 convolution layer and 32 convolution filters. Within 10 epochs, it has surpassed the best of my MLP model by 1%. The conv layer might be able to construct some important features from the image.

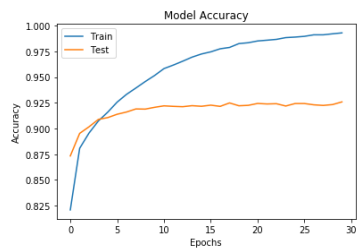


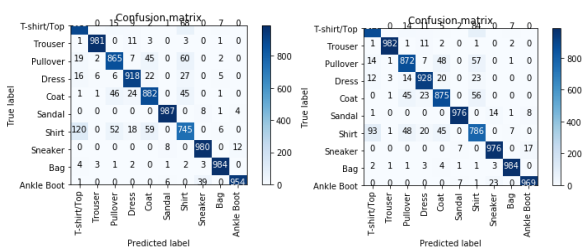




2. Pooling layer. The idea of a pooling layer is to compress the image while preserve it's feature, so that the later conv layers can recognize features in larger scale without increase its filter size. However, the images in this dataset is already highly compressed, my hypothesis is that further compression may cause feature lost. To support this hypothesis, the second experiment would be a comparison between a model that uses pooling vs a model with larger conv filters. As shown in the experiment 2, both models produce identical test accuracy, which is 92% for accuracy, but the larger conv model has a twice higher loss rate of 0.41 and takes almost twice longer to train each epoch. Therefore, the hypnosis is busted, as the larger conv filters does not bring positive impact to the training result, but it only brings negative impact to the training speed. And one pooling layers does not bring negative impact to the features at all. The reason of it could be that the 32 different conv filters is looking at the image in different perspective and have captured a good portion of the features, therefore the information of the feature is not lost in the compression.

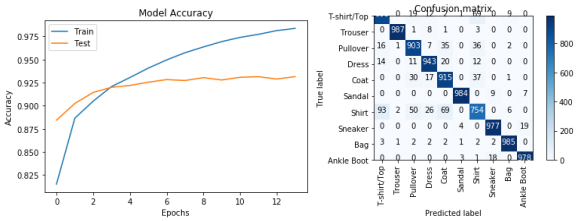




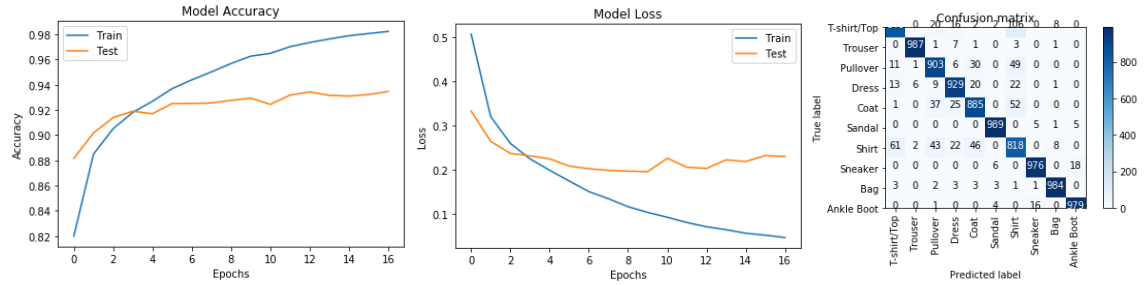


3. Compressing limitations. As one layer of pooling + conv have brought a positive impact to the test result, but it also shrinks the image by half, the hard limitation of this procedure would be when the image is compressed down to 1*1, but to what level of compression would this procedure be beneficial? As demonstrated in the 3rd experiment, the second pool-conv combination is not brining positive impact as it has similar test results. And the third pool-conv combination have lowered the accuracy by 1% Which indicates that the image should not be pooled to under 14*14.

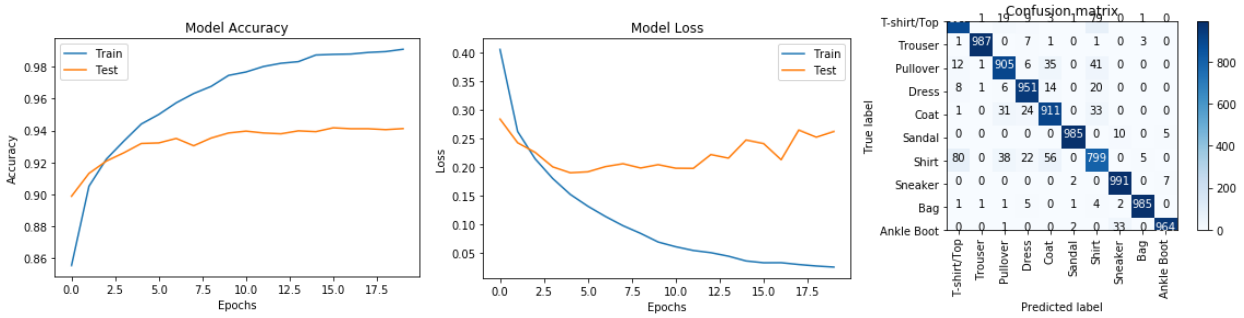
4. Number of conv layers. The 4th experiment will explore the impact of more conv layers. Therefore, one conv layer is added both before and after the pooling layer. the model reaches 93% of validation accuracy after the 9th epoch and have a test accuracy at 93.14% as well.



5. Dropout. As shown in the experiment 4, the model is highly overfitted, therefore this experiment will attempt to reduce the overfitting by adding dropout layers. The peak validation accuracy of this model has reached 93.47% around 17th epoch, but the test accuracy is only 92.96%. the peak performance is increased, but the rate of overfitting is not reduced at all.



6. Batch normalization. In the experiments of MLP, normalizing the data have brought a performance boost to the model, in the CNN experiments, the input to the conv layers are already normalized, but the product of the conv layers is not, therefore the 6th experiment will attempt to add batch normalization besides the flatten layer. The results are quite satisfying, as the validation accuracy have reached 94.17% for the first time and the testing accuracy is also reaching 93.65%. This performance is unprecedented in the benchmark of this dataset [1] for normal CNN nets without special structure and no augmentation to the dataset. Above that, the model also reaches such performance in only 20 epochs. At this point I would like to conclude the margins left to explore under this set of constrains are very limited.



iii. Decision tree

There are three main parameters to tune for decision tree, the criterion, max depth and splitter. As shown in the A2-decision tree notebook they all have around 79%-81% of accuracy. The Gini criterion tends to reduce training time comparing to the entropy but may bring slightly worse training outcome, the random splitter also reduces the training time comparing to the best splitter. And a deeper tree will take more time to train. It also appears that the deeper the tree is, the rate of overfitting is higher. Because unlike the networks, an unlimited decision tree could fit

the entire training set to an accuracy of 100%, however such a tree would have very low performance on unseen data due to overfitting.

iv. Random forest

Based on the findings from the experiment on Decision trees, I'm not going to attempt the Gini criterion as it may cause negative impact to the training outcome. The current best performing decision tree model that listed in the benchmark page is using entropy criterion and 100 estimators with max depth of 50, it has a accuracy of 88%. I decide to use it as a starting point and experiment that what impact some changes may bring to the result. Unlike the regular decision tree, limiting the depth to 10 brought negative impact of 3% to the accuracy of the model. it seems a random forest don't suffer from overfitting as much as decision trees due to its bagging strategy. Doubling the number for estimators and the depth also did not bring major negative impact to the model, it seems the decision tree is a rather stable model due to its voting mechanism, and it will not over fit the model when more resource then necessary is given.

c. Results & evaluation

i. Training cost

In the year of 2019, with up to date hardware and properly designed models, the training cost for such a toy dataset is less of a concern. The MLP final model costs 2s*100 epochs to train on my computer as it is only a simple model has one single hidden layer of 100 nodes. For the CNN final model, it took 9s*20 epochs to reach its prime, which is also not particularly costly to train. The training time of the decision tree is particularly less than 20 second and the random forest will also finish its training within 1-2 minutes.

III. Findings & Ultimate judgement

For the task of classifying the test dataset of fashion MNIST, I would recommend using the CNN model, due to it is highly accurate and not the costliest to train. However, for classifying real world image of clothes, none of the model here would be very useful, since its training dataset only consists of front centred images of the clothes, that is probably also wore by the same type of mannequin, which would make the model extremely hard to generalize, in this case, augmenting the dataset would not help too much as one entire dimension of information is completely lost, there are no way any data augmentation could generate the image of the objects in its top or side perspective. Above all, the dataset itself is trying too hard to become a drop-in replacement of the MNIST handwriting dataset, as the real world clothing's has far more categories than the ones in the dataset and the dataset also drops the second most important feature of the clothes, which is its colours. In conclusion, the training outcome may be useful for some sort of knowledge transfer to a newer model that trained in a much larger dataset but using them to classify real world clothes would bring poor performance.

IV. Reference

1. GitHub page of fashion MNIST dataset https://github.com/zalandoresearch/fashion-mnist#why-we-made-fashion-mnist
2. Benchmark page of fashion MNIST dataset https://github.com/zalandoresearch/fashion-mnist#why-we-made-fashion-mnist
3. Kong, Y. and Yu, T., 2018. A deep neural network model using random forest to extract feature representation for gene expression data classification. *Scientific reports*, *8*(1), p.16477. https://www.nature.com/articles/s41598-018-34833-6
4. Shotton, Jamie, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. "Real-time human pose recognition in parts from single depth images." In CVPR 2011, pp. 1297-1304. Ieee, 2011. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/BodyPartRecognition.pdf