

## Classification on Fashion MNIST Dataset

### Problem

Image classification is a classic question for deep learning. In this project, I will try to build a model to classify fashion MNIST data into 10 classes. This project is a practice on an advanced dataset than MNIST(10 categories of handwritten digits), the final model can serve as the starting point for image classification project on similar dataset.

Potential clients of this project can be: fashion website which is looking for a classification model to automatically classify fashion images; anyone who wants to build a classification model on similar dataset such as MNIST and more complex dataset with appropriate modification on data and model; and deep learning beginner who wants to get started with image classification.

### Data

#### Data Overview

Fashion MNIST dataset consists of 60,000 gray-scaled images for training and 10,000 images for testing. Each image size is 28x28 pixels, that is, 784 features with each feature as a value from 0 to 255 (the pixel intensity, 0 for white and 255 for black). And each image is associated with a label from 10 labels below:

<b><i>Label</i></b>	<b><i>Description</i></b>
0	<i>T-shirt/top</i>
1	<i>Trouser</i>
2	<i>Pullover</i>
3	<i>Dress</i>
4	<i>Coat</i>
5	<i>Sandal</i>

- 6      *Shirt*
- 7      *Sneaker*
- 8      *Bag*
- 9      *Ankle boot*

Each label has 6000 images in training set.

Let's take a look at first 9 images in training set with their label:



## Feature Scaling

Each feature has a value in the range of 0 to 255, which is too wide. Therefore, normalization is necessary to scale the data dimensions to make them approximately the same scale.

In this case, I scaled these values to a range of 0 to 1 by dividing them by 255.

# Modeling

## Classification using Machine Learning Methods

It is a multi-class classification problem. We need to classify images into 10 classes. Various methods are applicable in this case, below are 3 models I tried:

- SGD Classifier
- Decision Tree Classifier
- Random Forest Classifier

Metrics used to compare performance are accuracy, precision, recall and F1 score, and 5-fold cross validation was applied to calculate metrics, in case of overfitting problem.

Table 1: Machine Learning Classification Results

	SGD Classifier	Decision Tree	Random Forest
Accuracy	0.864	1.000	0.995
Precision	0.865	1.000	0.995
Recall	0.864	1.000	0.995
F1	0.865	1.000	0.995
CV Accuracy	0.839	0.796	0.859
CV Precision	0.838	0.797	0.858
CV Recall	0.839	0.796	0.859
CV F1	0.837	0.796	0.857

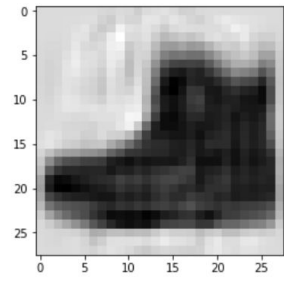
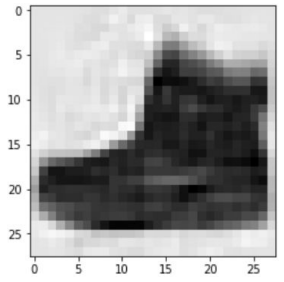
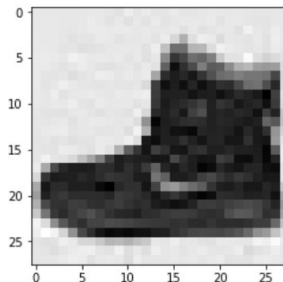
According to the model scores displayed above, we can see that Decision Trees Classifier and Random Forest Classifier both had a high accuracy on training data, but lower CV accuracy score. Obviously both models are overfitting, since Random Forest Classifier did a better work on cross-validation score, I continued to fine tuning the Random Forest Model.

Each image is of size 28\*28 which means 784 pixels in total. In order to speed up the model fitting process, I applied Principal Component Analysis(PCA) to reduce the dimension of input data.

Below is a table that shows the amount of features left after PCA and an example of image recovered when n\_components takes different values. When n\_components takes a value between 0 and 1, it represents the percentage of minimum amount of variance that needs to be explained, in this case, 90%, 95% and 99% respectively.

Under n\_components=0.9 and 0.95 we lose ~90% and ~75% features, the quality of image is poor, so 0.99 is a good value for n\_components.

Table 2: Dimension Reduction by PCA

<b>N_components =</b>	<b>0.9</b>	<b>0.95</b>	<b>0.99</b>
# of features left	84 (11.0%)	187 (24.0%)	459 (59.0%)
Sample of image recovered			

After PCA, I tuned the below hyperparameters with Grid Search technique:

- `n_estimators`: number of estimators in the model, values [50, 100]
- `max_depth`: maximum depth of trees, values [10, 20, 50]

Table 3: Grid Search CV results

<b>max_depth</b>	<b>n_estimators</b>	<b>Accuracy</b>
10	50	0.817
10	100	0.822
20	50	0.841
20	100	0.850
50	50	0.841
<b>50</b>	<b>100</b>	<b>0.852</b>

Table 3 shows the cross-validation accuracy score on training set under different values of `max_depth` and `n_estimators`. We can see that by increasing `n_estimators` and `max_depth`, we can get better scores from the model, however, there is overfitting problem at the same time:

Table 4: Scores of the best Random Forest Classifier on training set and test set

	<b>Accuracy</b>	<b>Precision</b>	<b>Recall</b>	<b>F1</b>
<b>Training set</b>	1.0	1.0	1.0	1.0
<b>Test set</b>	0.849	0.847	0.849	0.846

## Classification using Deep Learning Methods

Deep Learning is another approach to deal with this multi-class classification problem. Convolutional Neural Networks (CNNs) is a good neural network model to be used in this case.

### Convolutional Neural Networks(CNNs)

I created several CNN based models to classify the images from Fashion MNIST training set, and evaluated each model's performance. I built the model with Keras framework. Here are the list of models I tried out on this dataset:

- CNN with 1 Convolutional Layer
- CNN with 2 Convolutional Layers
- CNN with 3 Convolutional Layers

Here is my approach:

1. Before building the models, I split the original training set into training(80%) and validation(20%) sets. Training set was used to train the model and validation set was used for optimizing the model. Validation set can also help us detect overfitting problem.

Test set was kept original to test the model performance on unseen data.

2. Each model was compiled with categorical crossentropy loss function and Adam optimizer after all layers have been added.
3. Each model was trained for 15 epochs, with batch size of 64
4. Each model was evaluated by accuracy, on training set, validation set, and test set.

Table 5: CNNs with scores

<b>CNN Model</b>	<b>Accuracy (Train)</b>	<b>Accuracy (Validation)</b>	<b>Accuracy (Test)</b>
1 convolutional layer, 1 max pooling layer	0.948	0.922	0.914
2 convolutional layers, 2 max pooling layers	0.906	0.911	0.907
3 convolutional layers, 2 max pooling layers	0.949	0.931	0.928

The CNN model with 3 convolutional layer has the highest accuracy, the accuracy on the test set is 92.8% which is much higher than that of Random Forest Classifier. According to the score on train set and validation set, the model is still a little bit overfitting, but acceptable.

## Transfer Learning

Transfer Learning can be an efficient approach to deep learning such kind of data. By using a pre-trained network which has been trained on a large scaled data set, time and effort can be saved in training process. In addition, these pre-trained networks can sometimes provide knowledge and hierarchy of features that can not be learnt from small dataset.

In this project, I used VGG19 model with weights learnt from Imagenet, to train the model on our Fashion MNIST data.

Here is my approach:

1. Created a base model of VGG19, without the 3 fully-connected layers at the top of the network. Set the input shape as (100, 100, 3) (100pixel \* 100pixel with 3 channels), classes as 10 (10 classes of labels).
2. Converted the data into the shape that was expected by the model. The original training set was of shape (60000, 28, 28), after reshaping and resizing, the shape turned out to be (60000, 100, 100, 3).
3. Split the original training set into training(80%) and validation(20%) sets.
4. Preprocessed the input specifically for VGG19 using `keras.applications.vgg19.preprocess_input()` function
5. Extracted features from VGG19 by letting VGG19 make predictions on the training set, validation set and test set.
6. Added densely connected classifier followed by a LeakyReLU layer and final dense layer for the number of classes to a Sequential model in Keras framework, compiled the model with categorical crossentropy loss function and Adam optimizer after all layers have been added, and used the extracted features from VGG19 to train the model.
7. The model was evaluated by the accuracy on training, validation and test set.

Table 6: Transfer Learning Result

	<b>Accuracy (Train)</b>	<b>Accuracy (Validation)</b>	<b>Accuracy (Test)</b>
VGG19	0.330	0.328	0.326

## Final Model

The model which has the highest prediction accuracy is the CNN model with 3 convolutional layers.

Below are the details of the model:

Model: "CNN with 3 convolutional layers"

Layer (type)	Output Shape	Param #
conv2d_26 (Conv2D)	(None, 26, 26, 32)	320
conv2d_27 (Conv2D)	(None, 24, 24, 64)	18496
max_pooling2d_19 (MaxPooling)	(None, 12, 12, 64)	0
dropout_20 (Dropout)	(None, 12, 12, 64)	0
conv2d_28 (Conv2D)	(None, 10, 10, 128)	73856
max_pooling2d_20 (MaxPooling)	(None, 5, 5, 128)	0
dropout_21 (Dropout)	(None, 5, 5, 128)	0
flatten_11 (Flatten)	(None, 3200)	0
dense_21 (Dense)	(None, 128)	409728
dense_22 (Dense)	(None, 10)	1290
Total params: 503,690		
Trainable params: 503,690		
Non-trainable params: 0		

In order to better evaluate the model performance, here is the classification report:

	precision	recall	f1-score	support
Class 0: T-shirt/top	0.87	0.88	0.88	1000
Class 1: Trouser	0.99	0.99	0.99	1000
Class 2: Pullover	0.91	0.89	0.90	1000
Class 3: Dress	0.94	0.92	0.93	1000
Class 4: Coat	0.88	0.91	0.90	1000
Class 5: Sandal	0.99	0.98	0.99	1000
Class 6: Shirt	0.79	0.77	0.78	1000
Class 7: Sneaker	0.95	0.98	0.97	1000
Class 8: Bag	0.98	0.99	0.99	1000
Class 9: Ankle boot	0.98	0.96	0.97	1000

accuracy			0.93	10000
macro avg	0.93	0.93	0.93	10000
weighted avg	0.93	0.93	0.93	10000

We can see that the model is not good at classify Shirt and T-shirt/top. As the scores of both classes are relatively lower than that of other classes.

Below are some examples of images that have been incorrectly classified:



Some images are difficult to classify even for human beings!



## Conclusion

Fashion MNIST is a good starting point for Machine Learning and Deep Learning beginners.

By Machine Learning models, I got a classification accuracy of 85% on the test data using Random Forest Classifier to predict.

By Deep Learning methods, the best model CNN with 3 convolutional layers reaches an accuracy of 93% on the test data!

## Limitations and Next Steps

The CNN model with 3 convolutional layers is underperforming for class 0 T-shirt/Top and class 6 Shirt. Next step can be a model that can learn more from mistakes, therefore improve the accuracy on the two classes mentioned above.

Due to the capacity of my device, I was only able to process data in a small batch size, increasing the batch size may improve the model trained, but will definitely be more time and resource consuming. Possible solutions can be trying out more pre-trained models and running model training on GPUs.