**Contents**

1. [Introduction](https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist#1)
2. Architecture
3. [Data exploration](https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist#4)
4. [Class distribution](https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist#41)
5. [Images samples](https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist#42)
6. [Model](https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist#5) building & training
7. [Test prediction accuracy](https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist#53)
8. [Validation accuracy and loss](https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist#54)
9. [Visualize the classified images](https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist#6)
   1. [Correctly classified images](https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist#61)
   2. [Incorrectly classified images](https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist#62)
10. [Conclusions](https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist#7)

Introduction:

## **Dataset**

Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. Zalando intends Fashion-MNIST to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits.

## **Content**

Each image is 28 pixels in height and 28 pixels in width, for a total of 784 pixels in total.

Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255.

The training and test data sets have 785 columns.

The first column consists of the class labels (see above), and represents the article of clothing.

The rest of 784 columns (1-785) contain the pixel-values of the associated image.

Architecture:



1. Define and understand the problem.
2. Download the data, and explore it for the insights.
3. Clean and preprocess for the best model accuracy.
4. Build the model
5. Measure the accuracy scores
6. Tune the model, by adding more layers,etc.
7. The models used to compare are tuned using hyperparameters.

Data Exploration:

There are 10 different classes of images, as following:

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

Image dimmensions are **28**x**28**.

The train set and test set are given in two separate datasets.

Fashion MNIST train - rows: 60000 columns: 785

Fashion MNIST test - rows: 10000 columns: 785

Class Distribution:

Train Data Set:

Ankle Boot : 6000 or 10.0%

Bag : 6000 or 10.0%

Sneaker : 6000 or 10.0%

Shirt : 6000 or 10.0%

Sandal : 6000 or 10.0%

Coat : 6000 or 10.0%

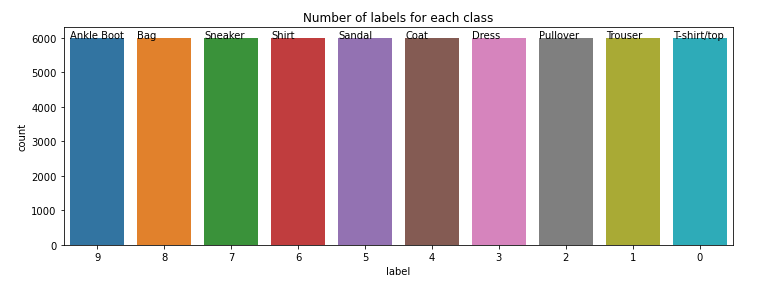
Dress : 6000 or 10.0%

Pullover : 6000 or 10.0%

Trouser : 6000 or 10.0%

T-shirt/top : 6000 or 10.0%

The classes are equaly distributed in the train set (10% each).[¶](http://localhost:8890/notebooks/Downloads/ineuron/fmnist%20.ipynb#The-classes-are-equaly-distributed-in-the-train-set-(10%-each).)



**Test Dataset:**

Sneaker : 1000 or 10.0%

Shirt : 1000 or 10.0%

Sandal : 1000 or 10.0%

Coat : 1000 or 10.0%

Dress : 1000 or 10.0%

Pullover : 1000 or 10.0%

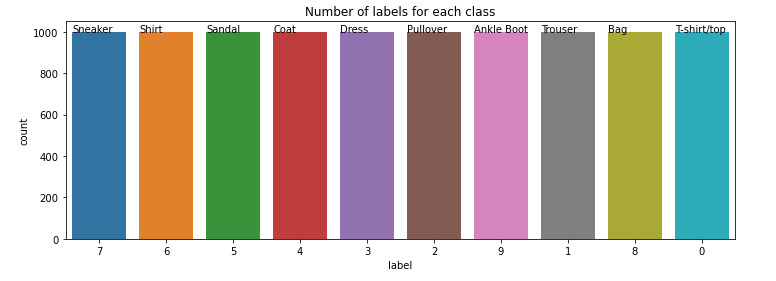
Ankle Boot : 1000 or 10.0%

Trouser : 1000 or 10.0%

Bag : 1000 or 10.0%

T-shirt/top : 1000 or 10.0%

The classes are equaly distributed in the train set (10% each).



**Sample Images**

**Train DataSet:**



**Test DataSet:**

****

**Model building and Training**

1. Dataset is read from the given location
2. Exploratory Data analysis was performed to check the classification of the data set. .
3. Data Preprocessing
   1. First we will do a data preprocessing to prepare for the model.
   2. We reshape the columns from (784) to (28,28,1). We also save label (target) feature as a separate vector.
   3. The dimmension of the processed train, validation and test set are as following:
4. Split train in train and validation set

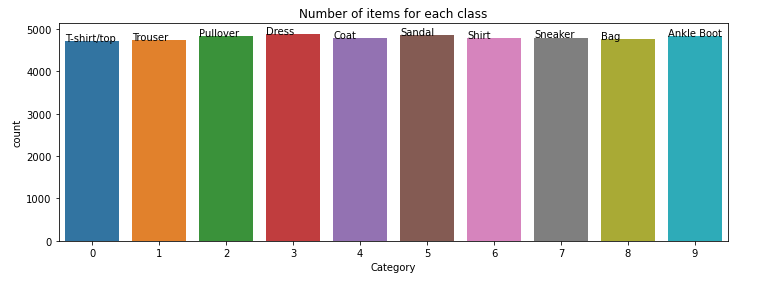
We further split the train set in train and validation set. The validation set will be 20% from the original train set, therefore the split will be train/validation of 0.8/0.2.

Fashion MNIST train - rows: 48000 columns: (28, 28, 1)

Fashion MNIST valid - rows: 12000 columns: (28, 28, 1)

Fashion MNIST test - rows: 10000 columns: (28, 28, 1)

## class distribution for the resulted training set.



## Dress : 4891 or 10.189583333333333%

## Sandal : 4855 or 10.114583333333334%

## Pullover : 4836 or 10.075000000000001%

## Ankle Boot : 4827 or 10.05625%

## Sneaker : 4798 or 9.995833333333334%

## Coat : 4796 or 9.991666666666667%

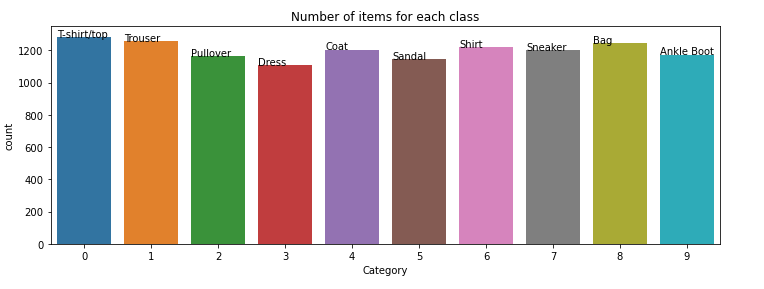
## Shirt : 4779 or 9.95625%

## Bag : 4757 or 9.910416666666666%

## Trouser : 4745 or 9.885416666666668%

## T-shirt/top : 4716 or 9.825000000000001%

## class distribution for the resulted validation set.



T-shirt/top : 1284 or 10.7%

Trouser : 1255 or 10.458333333333334%

Bag : 1243 or 10.358333333333333%

Shirt : 1221 or 10.174999999999999%

Coat : 1204 or 10.033333333333333%

Sneaker : 1202 or 10.016666666666667%

Ankle Boot : 1173 or 9.775%

Pullover : 1164 or 9.700000000000001%

Sandal : 1145 or 9.541666666666666%

Dress : 1109 or 9.241666666666665%

1. Model :
   1. We will use a Sequential model.
   2. The Sequential model is a linear stack of layers. It can be first initialized and then we add layers using add method or we can add all layers at init stage. The layers added are as follows:
   3. For activation : softmax; for this final layer it is used softmax activation (standard for multiclass classification)
   4. Then we compile the model, specifying as well the following parameters:

1.loss

2.optimizer

3.metrics.

* 1. Run the model with the training set.
  2. Also using the validation set (a subset from the orginal training set) for validation.

[**Test prediction accuracy**](https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist#53)**:**

Test loss: 0.3846653401851654

Test accuracy: 0.8907999992370605

Changed the Dense parameters and retrained the model and tested for accuracy

Test loss: 0.3887470066547394

Test accuracy: 0.8984000086784363

**Validation accuracy:**

Correct predicted classes: 8908

Incorrect predicted classes: 1092

precision recall f1-score support

Class 0 (T-shirt/top) : 0.81 0.87 0.84 1000

Class 1 (Trouser) : 0.98 0.98 0.98 1000

Class 2 (Pullover) : 0.80 0.81 0.81 1000

Class 3 (Dress) : 0.92 0.89 0.90 1000

Class 4 (Coat) : 0.79 0.88 0.84 1000

Class 5 (Sandal) : 0.97 0.95 0.96 1000

Class 6 (Shirt) : 0.79 0.64 0.71 1000

Class 7 (Sneaker) : 0.93 0.96 0.94 1000

Class 8 (Bag) : 0.96 0.98 0.97 1000

Class 9 (Ankle Boot) : 0.96 0.95 0.95 1000

accuracy 0.89 10000

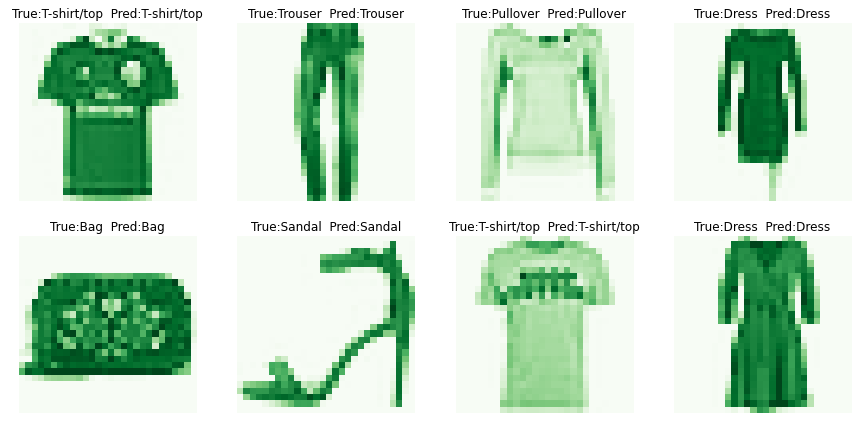
macro avg 0.89 0.89 0.89 10000

weighted avg 0.89 0.89 0.89 10000

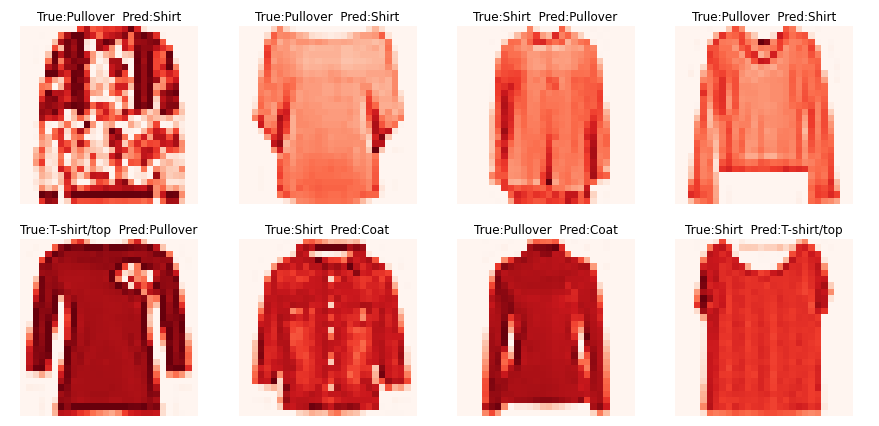
* The best accuracy is obtained for Class 1, Class 5, Class 8, Class 9 and Class 7.
* Worst accuracy is for Class 6.
* The recall is highest for Class 8, Class 5 and smallest for Class 6 and Class 4.
* f1-score is highest for Class 1, Class 5 and Class 8
* smallest for Class 6 followed by Class 4 and Class 2

[**Visualize the classified images**](https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist#6)

* 1. [Correctly classified images](https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist#61)



* 1. [Incorrectly classified images](https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist#62)



[**Conclusions**](https://www.kaggle.com/gpreda/cnn-with-tensorflow-keras-for-fashion-mnist#7)

Only few classes are not correctly classified all the time, especially Class 6 (Shirt) and Class 2 (Pullover).