# Experimentation of CNN with Fashion MNIST Dataset

## Prequisites

* Tensorflow 1.12
* Keras
* Python 3.x

## Dataset

[Fashion-MNIST](https://github.com/zalandoresearch/fashion-mnist) is 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.

## How to Use

1. Open [this repository notebook](https://colab.research.google.com/drive/1o_8lMv4jBnYIwOVeGu106k_J1lc5-F7y)
2. Run each step by pressing “play” button or CTRL + ENTER
3. Change the parameters in “Parameters Section”

## Run the Models

1. Go to section [running models](https://colab.research.google.com/drive/1o_8lMv4jBnYIwOVeGu106k_J1lc5-F7y#scrollTo=vF6OkgT7fmbk)
2. Select model that want to evaluate
3. Run evaluate

## Training Steps

### Importing required modules

import tensorflow as tf  
from tensorflow import keras  
from sklearn.model\_selection import StratifiedShuffleSplit  
from keras import backend as k  
  
import numpy as np  
import matplotlib.pyplot as plt

### Load Datasets

(imageTrain, labelTrain), (imageTest, labelTest) = tf.keras.datasets.fashion\_mnist.load\_data()

### Make array of labels and split training data into validation data

labels = {0 : "T-shirt/top", 1: "Trouser", 2: "Pullover", 3: "Dress", 4: "Coat", 5: "Sandal", 6: "Shirt", 7: "Sneaker", 8: "Bag", 9: "Ankle Boot"}  
  
imageTrain = np.expand\_dims(imageTrain, -1)  
imageTest = np.expand\_dims(imageTest, -1)  
  
sss = StratifiedShuffleSplit(n\_splits=5, random\_state=0, test\_size=1/6)  
trainIndex, validIndex = next(sss.split(imageTrain, labelTrain))  
imageValid, labelValid = imageTrain[validIndex], labelTrain[validIndex]  
imageTrain, labelTrain = imageTrain[trainIndex], labelTrain[trainIndex]

### Perform Mean Subtraction and Normalization of Datasets (a)

* Scaling down range of input value into between 0 and 1 instead of 0 - 255

imageTrain = imageTrain / 255  
imageValid = imageValid / 255  
imageTest = imageTest / 255

* Mean Substraction of data

meanSubt = np.mean(imageTrain)  
  
imageTrain = imageTrain - meanSubt  
imageValid = imageValid - meanSubt  
imageTest = imageTest - meanSubt

* Normalization

stdDev = np.std(imageTrain)  
  
imageTrain = imageTrain / stdDev  
imageValid = imageValid / stdDev  
imageTest = imageTest / stdDev

Mean subtraction and normalization using image training data to make sure it’s same across all datasets that we use. This process necessary because we want to make our data sparse before we train.

### Choosing Initializer (b)

We experiment with two weight initialization: [He initialization](https://arxiv.org/abs/1502.01852) and [Xavier Initialization](http://proceedings.mlr.press/v9/glorot10a.html)

We can see the result in below table:

|  |  |  |
| --- | --- | --- |
|  | Xavier Initializer | He Initializer |
| Accuracy | 0.9663 | 0.9675 |
| Loss | 0.0891 | 0.0863 |
| Validation Accuracy | 0.9194 | 0.9184 |
| Validation Loss | 0.284 | 0.2929 |
| Test Accuracy | 0.9138 | 0.9159 |
| Test Loss | 0.3143 | 0.3164 |

### Experimentation with network configuration (c)

|  |  |  |
| --- | --- | --- |
|  | 32 Filter + 2 Layers | 64 Filter + 2 Layers |
| Accuracy | 0.9663 | 0.9675 |
| Loss | 0.0891 | 0.0863 |
| Validation Accuracy | 0.9194 | 0.9184 |
| Validation Loss | 0.284 | 0.2929 |
| Test Accuracy | 0.9138 | 0.9159 |
| Test Loss | 0.3143 | 0.3164 |

### Experimentation with gradient optimization techniques (d)

### Experimentation with activation functions (e)

### Experimentation with regularization techniques (f)