fashion2

September 4, 2019

##CREATION OF CNN MODEL FOR FASHION MNIST DATASET

0.0.1 Load fashion mnist dataset using keras

```
[1]: import tensorflow as tf
     fmnist = tf.keras.datasets.fashion_mnist

     (x_train, y_train),(x_test, y_test) = fmnist.load_data()
```

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/train-labels-idx1-ubyte.gz
32768/29515 [==================================] - 0s 1us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 7s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/t10k-labels-idx1-ubyte.gz
8192/5148 [===] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/t10k-images-idx3-ubyte.gz
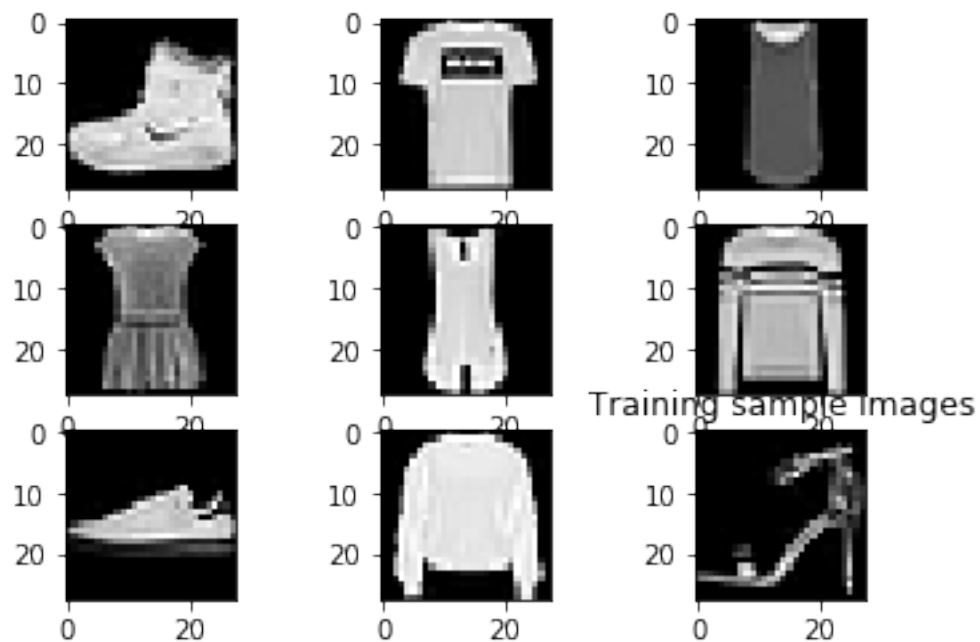4423680/4422102 [=============================] - 1s 0us/step

###Sample Fashion Mnist Training images

```
[3]: from matplotlib import pyplot as plt


     for i in range(9):
       plt.subplot(330 + 1 +i)
       plt.imshow(x_train[i],cmap=plt.get_cmap('gray'))


     plt.title('Training sample images')
```
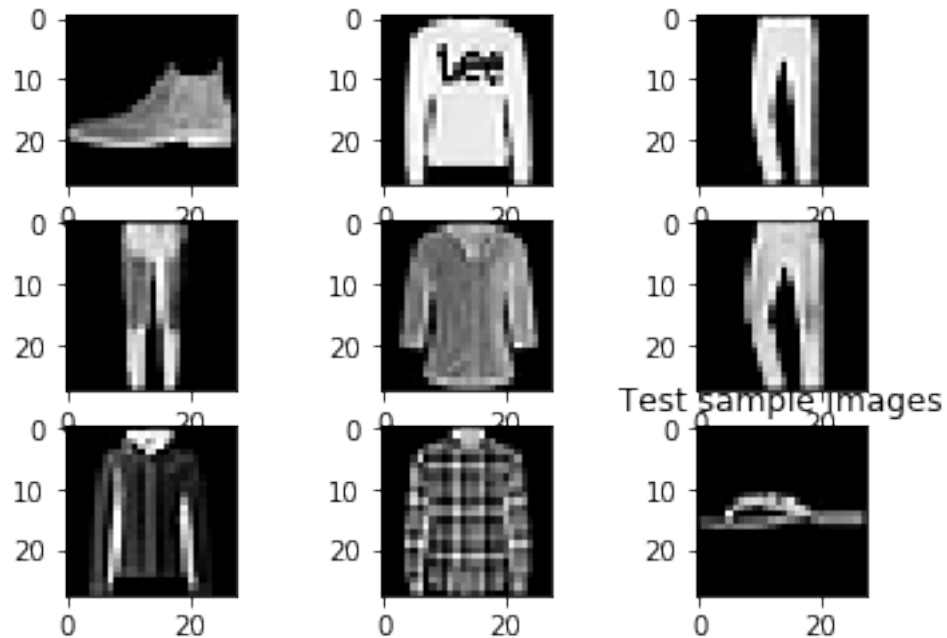
```
[3]: Text(0.5, 1.0, 'Training sample images')
```

Training sample images

Sample Testing images

```python
[4]: from matplotlib import pyplot as plt


for i in range(9):
    plt.subplot(330 + 1 +i)
    plt.imshow(x_test[i],cmap=plt.get_cmap('gray'))


plt.title('Test sample images')
```

[4]: Text(0.5, 1.0, 'Test sample images')

Test sample images

###Pre-processing of Training and Testing set

0.0.2 Tensorflow model uses four dimensions

0.0.3 image size of fashion mnist is 28 x28

```
[5]: x_train=x_train.reshape(x_train.shape[0],28,28,1)
     x_test=x_test.reshape(x_test.shape[0],28,28,1)
     ## convert to floating point to process the data with tensorflow
     x_train=x_train.astype('float32')
     x_test=x_test.astype('float32')
     ## Normalization of gray-scale image
     x_train/=255;
     x_test/=255;
```

```
[6]: ##displaying the size of training and testing set
     print('Training set', x_train.shape)
     print('Testing set', x_test.shape)
```

```
Training set (60000, 28, 28, 1)
Testing set (10000, 28, 28, 1)
```

###preparation of class labels using hot-one encoding technique

```
[7]: from keras.utils import np_utils
     y_train=np_utils.to_categorical(y_train)
     y_test=np_utils.to_categorical(y_test)
     num_classes=y_test.shape[1]
```

```python
print('Training set labels', y_train.shape)
print('Testing Set labels', y_test.shape)
print('Number of classes',num_classes)
```

```
Training set labels (60000, 10)
Testing Set labels (10000, 10)
Number of classes 10
```

```
Using TensorFlow backend.
```

###Creation of CNN model
###Convolution layers - 2 filter size 3x3, Subsampling using Maxpooling layer-1, Regularization using Dropout, Activation function -RELU

0.0.4 Fully Connected layers-3 and the last layer is FC layer using Activation function SOFT-MAX which is a logistic regression method which assigns the class labels

```python
[8]: import keras
from keras.models import Sequential
from keras.layers import Dense,Dropout, Flatten
from keras.layers import Conv2D,MaxPooling2D
from keras import backend as K
from keras.optimizers import SGD
from keras.optimizers import Adam
model =Sequential()
input_shape=(28,28,1)
model.add(Conv2D(32,kernel_size=(3, 3),activation='relu',
  →input_shape=input_shape))
model.add(Conv2D(64,kernel_size=(3, 3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.20))
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.20))

model.add(Dense(num_classes,activation='softmax'))
```

```
WARNING:tensorflow:From /home/user/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:66: The name tf.get_default_graph
is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From /home/user/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:541: The name tf.placeholder is
deprecated. Please use tf.compat.v1.placeholder instead.
```

```
WARNING:tensorflow:From /home/user/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:4432: The name tf.random_uniform is
deprecated. Please use tf.random.uniform instead.

WARNING:tensorflow:From /home/user/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:4267: The name tf.nn.max_pool is
deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From /home/user/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:148: The name
tf.placeholder_with_default is deprecated. Please use
tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /home/user/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:3733: calling dropout (from
tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be removed
in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 -
keep_prob`.
```

0.1 Model Compilation and display of model summary with all hyperparameters

0.1.1 Optimizer used for minimizing the global cost function is Adam optimizer with a learning rate of 0.008

```
[9]: model.compile(loss='categorical_crossentropy', optimizer=Adam(0.
      ↪0008),metrics=['accuracy'])

     print(model.summary())
```

```
WARNING:tensorflow:From /home/user/anaconda3/lib/python3.7/site-
packages/keras/optimizers.py:793: The name tf.train.Optimizer is deprecated.
Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /home/user/anaconda3/lib/python3.7/site-
packages/keras/backend/tensorflow_backend.py:3576: The name tf.log is
deprecated. Please use tf.math.log instead.

Model: "sequential_1"
_____
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_____
conv2d_2 (Conv2D)            (None, 24, 24, 64)        18496
_____
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 64)        0
```

```
    -------------------------------------------------------------------
    dropout_1 (Dropout)         (None, 12, 12, 64)         0
    -------------------------------------------------------------------
    flatten_1 (Flatten)         (None, 9216)               0
    -------------------------------------------------------------------
    dense_1 (Dense)             (None, 128)                1179776
    -------------------------------------------------------------------
    dropout_2 (Dropout)         (None, 128)                0
    -------------------------------------------------------------------
    dense_2 (Dense)             (None, 128)                16512
    -------------------------------------------------------------------
    dropout_3 (Dropout)         (None, 128)                0
    -------------------------------------------------------------------
    dense_3 (Dense)             (None, 10)                 1290
    ===================================================================
    Total params: 1,216,394
    Trainable params: 1,216,394
    Non-trainable params: 0
    -------------------------------------------------------------------
    None
```

##batch-size is set to 64 and number of epochs is 10

[10]:
```python
batchsize=32
epochs1=10
```

###Fitting the training set to model

[11]:
```python
results=model.
 →fit(x_train,y_train,batch_size=batchsize,epochs=epochs1,verbose=1,validation_data=(x_test,y_t
```

```
WARNING:tensorflow:From /home/user/anaconda3/lib/python3.7/site-
packages/tensorflow/python/ops/math_grad.py:1250:
add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is
deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Train on 60000 samples, validate on 10000 samples
Epoch 1/10
60000/60000 [==============================] - 220s 4ms/step - loss: 0.4693 -
acc: 0.8315 - val_loss: 0.3092 - val_acc: 0.8867
Epoch 2/10
60000/60000 [==============================] - 220s 4ms/step - loss: 0.2968 -
acc: 0.8921 - val_loss: 0.2719 - val_acc: 0.8997
Epoch 3/10
60000/60000 [==============================] - 233s 4ms/step - loss: 0.2476 -
acc: 0.9087 - val_loss: 0.2471 - val_acc: 0.9125
Epoch 4/10
60000/60000 [==============================] - 235s 4ms/step - loss: 0.2142 -
acc: 0.9206 - val_loss: 0.2248 - val_acc: 0.9186
```

```
Epoch 5/10
60000/60000 [==============================] - 235s 4ms/step - loss: 0.1834 -
acc: 0.9318 - val_loss: 0.2366 - val_acc: 0.9165
Epoch 6/10
60000/60000 [==============================] - 235s 4ms/step - loss: 0.1638 -
acc: 0.9392 - val_loss: 0.2303 - val_acc: 0.9225
Epoch 7/10
60000/60000 [==============================] - 234s 4ms/step - loss: 0.1484 -
acc: 0.9451 - val_loss: 0.2354 - val_acc: 0.9249
Epoch 8/10
60000/60000 [==============================] - 235s 4ms/step - loss: 0.1321 -
acc: 0.9493 - val_loss: 0.2412 - val_acc: 0.9244
Epoch 9/10
60000/60000 [==============================] - 235s 4ms/step - loss: 0.1221 -
acc: 0.9541 - val_loss: 0.2738 - val_acc: 0.9204
Epoch 10/10
60000/60000 [==============================] - 233s 4ms/step - loss: 0.1111 -
acc: 0.9584 - val_loss: 0.2790 - val_acc: 0.9193
```

0.1.2 Model Evaluation

```
[12]: score=model.evaluate(x_test,y_test,verbose=1)


      print('Test Loss',score[0])
      print('Test Accuracy',score[1])
```

```
10000/10000 [==============================] - 7s 749us/step
Test Loss 0.2790250390827656
Test Accuracy 0.9193
```
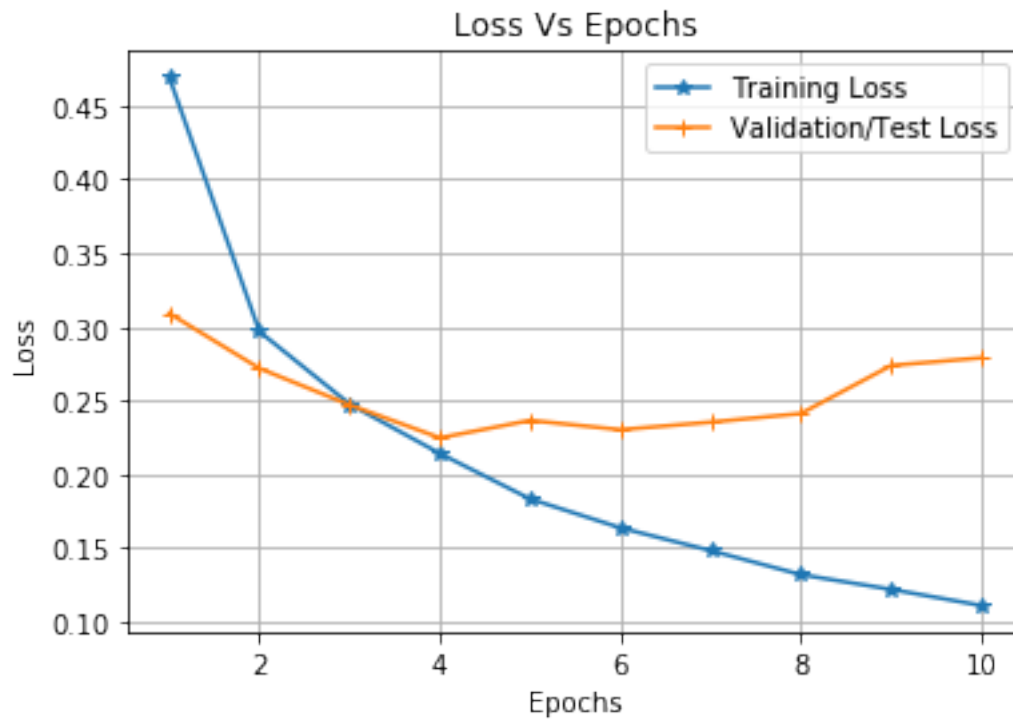
0.1.3 Graphical representation of Loss Vs Epochs

```
[13]: from matplotlib import pyplot as plt
      results1=results.history
      loss=results1['loss']
      validationloss=results1['val_loss']
      epochs=range(1,len(loss)+1)
      plt.plot(epochs,loss,label='Training Loss',marker='*')
      plt.plot(epochs,validationloss,label='Validation/Test Loss',marker='+')
      plt.title('Loss Vs Epochs')
      plt.xlabel('Epochs')
      plt.ylabel('Loss')
      plt.grid(True)
      plt.legend()
      plt.show()
```

Loss Vs Epochs

###Graphical representation of Accuracy Vs Epochs

```
[14]: from matplotlib import pyplot as plt
      results1=results.history
      training_accuracy=results1['acc']
      val_acc=results1['val_acc']
      epochs1=range(1,len(training_accuracy)+1)
      plt.plot(epochs1,training_accuracy,label='Training␣
       ↪Accuracy',marker="*",color='r')
      plt.plot(epochs1,val_acc,label='Validation Accuracy',marker="+",color='g')
      plt.title('Accuracy Vs Epochs')
      plt.xlabel('Epochs')
      plt.ylabel('Accuracy')
      plt.grid(True)
      plt.legend()
      plt.show()
```

Accuracy Vs Epochs
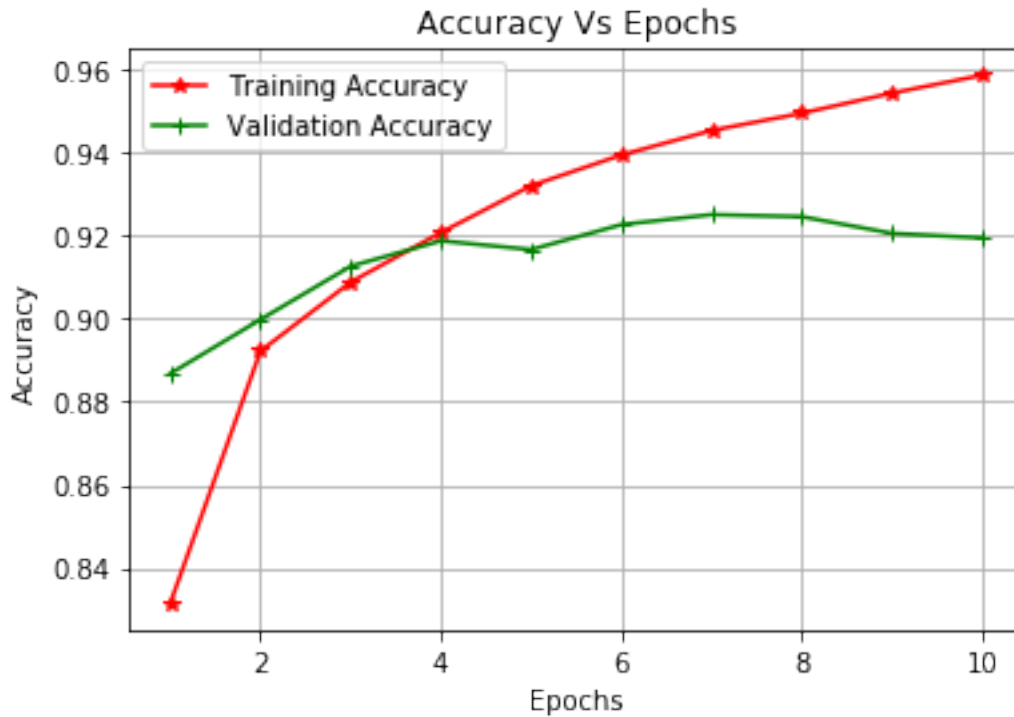
Display of Confusion matrix

```
[15]: from sklearn.metrics import confusion_matrix
      import numpy as np
      predicted=model.predict_classes(x_test)
      print(confusion_matrix(np.argmax(y_test,axis=1),predicted))
```

```
[[910    0  11  13    2    1  61    0    2    0]
 [   1 981    0  13    0    0    3    0    2    0]
 [ 17    1 848    9  56    0  67    0    2    0]
 [ 20    2    9 930  17    0  21    0    1    0]
 [   1    0  30  26 887    1  55    0    0    0]
 [   0    0    0    0    0 985    0  10    0    5]
 [140    0  36  25  52    0 735    0  12    0]
 [   0    0    0    0    0    5    0 982    0  13]
 [   4    0    1    3    2    1    2    2 985    0]
 [   0    0    0    0    0    8    0  42    0 950]]
```

[ ]: