mnist

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.mnist

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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/mnist.npz
11493376/11490434 [==============================] - 2s 0us/step
```

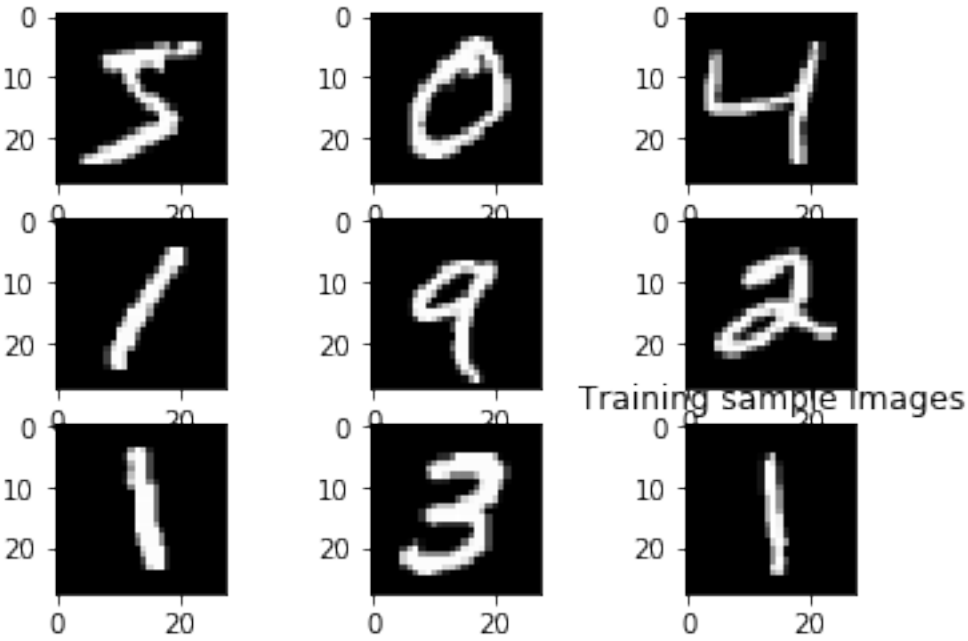
###Sample Fashion Mnist Training images

[4]:
```
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')
```

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



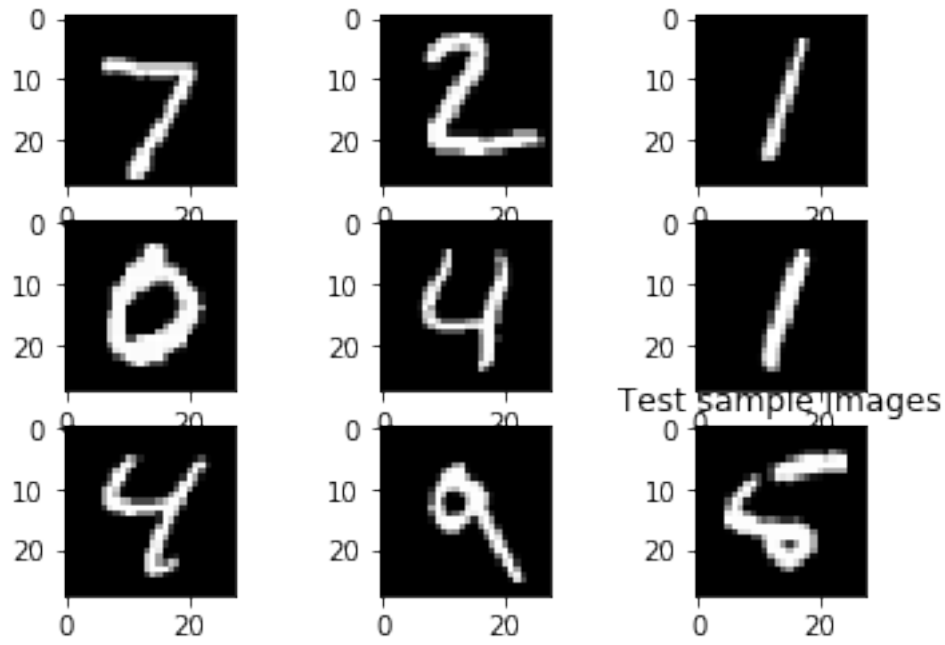
###Sample Testing images

[5]:
```
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')
```

[5]:
```
Text(0.5, 1.0, '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

[6]:
```
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;
```

[7]:
```
##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

[8]:
```
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]
```

```
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 SOFTMAX which is a logistic regression method which assigns the class labels

[9]:
```
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

[10]:
```
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

[11]:
```
batchsize=32
epochs1=10
```

###Fitting the training set to model

[12]:
```
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 [==============================] - 215s 4ms/step - loss: 0.1731 -
acc: 0.9468 - val_loss: 0.0574 - val_acc: 0.9813
Epoch 2/10
60000/60000 [==============================] - 219s 4ms/step - loss: 0.0612 -
acc: 0.9819 - val_loss: 0.0406 - val_acc: 0.9865
Epoch 3/10
60000/60000 [==============================] - 219s 4ms/step - loss: 0.0441 -
acc: 0.9868 - val_loss: 0.0374 - val_acc: 0.9884
Epoch 4/10
60000/60000 [==============================] - 229s 4ms/step - loss: 0.0346 -
acc: 0.9897 - val_loss: 0.0364 - val_acc: 0.9896
```

```
Epoch 5/10
60000/60000 [==============================] - 230s 4ms/step - loss: 0.0292 -
acc: 0.9912 - val_loss: 0.0306 - val_acc: 0.9904
Epoch 6/10
60000/60000 [==============================] - 215s 4ms/step - loss: 0.0226 -
acc: 0.9928 - val_loss: 0.0398 - val_acc: 0.9887
Epoch 7/10
60000/60000 [==============================] - 228s 4ms/step - loss: 0.0206 -
acc: 0.9936 - val_loss: 0.0339 - val_acc: 0.9906
Epoch 8/10
60000/60000 [==============================] - 229s 4ms/step - loss: 0.0170 -
acc: 0.9947 - val_loss: 0.0265 - val_acc: 0.9938
Epoch 9/10
60000/60000 [==============================] - 230s 4ms/step - loss: 0.0163 -
acc: 0.9949 - val_loss: 0.0361 - val_acc: 0.9913
Epoch 10/10
60000/60000 [==============================] - 229s 4ms/step - loss: 0.0135 -
acc: 0.9956 - val_loss: 0.0418 - val_acc: 0.9909
```

0.1.2 Model Evaluation

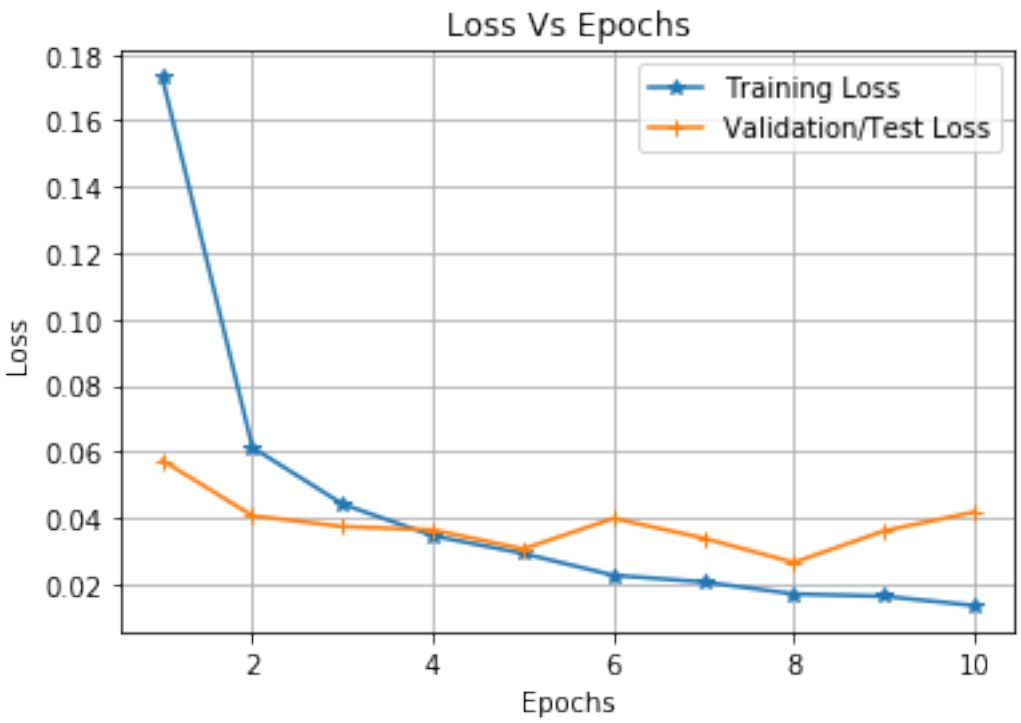
[13]:
```
score=model.evaluate(x_test,y_test,verbose=1)


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

```
10000/10000 [==============================] - 7s 748us/step
Test Loss 0.041754518986355876
Test Accuracy 0.9909
```

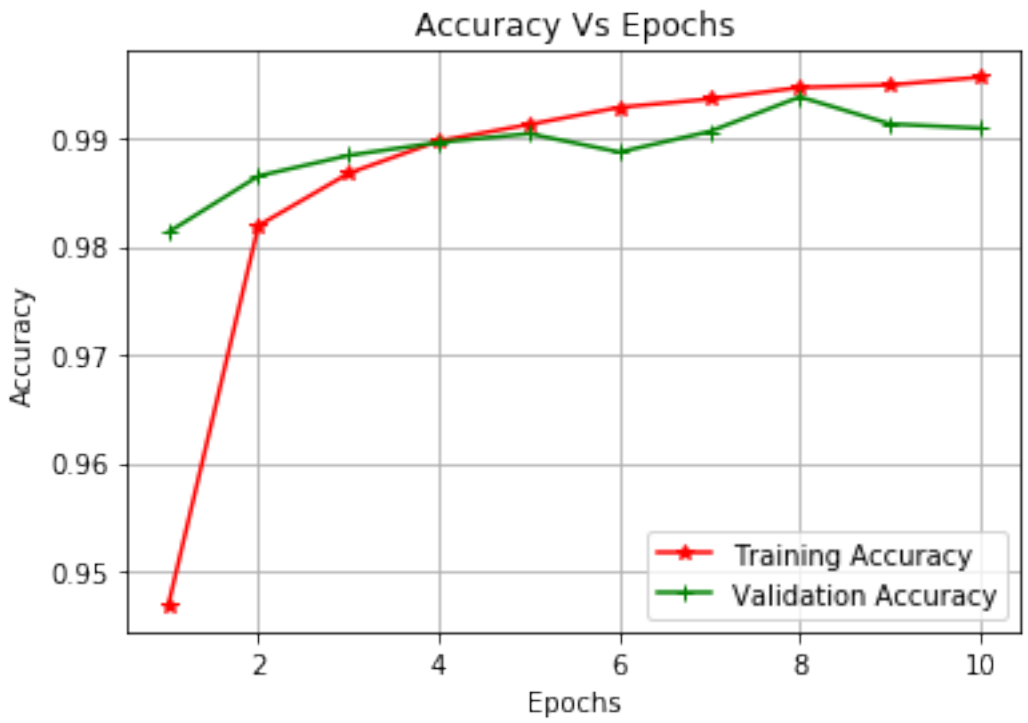
0.1.3 Graphical representation of Loss Vs Epochs

[14]:
```
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()
```



###Graphical representation of Accuracy Vs Epochs

[15]:
```
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()
```



###Display of Confusion matrix

[16]:
```
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))
```

```
[[ 979    0    0    1    0    0    0    0    0    0]
 [   0 1133    0    1    0    0    0    1    0    0]
 [   1    1 1018    2    1    0    0    9    0    0]
 [   0    0    0 1005    0    3    0    2    0    0]
 [   0    0    0    0  975    0    3    0    0    4]
 [   1    0    0    5    0  884    2    0    0    0]
 [   3    4    0    1    1    2  947    0    0    0]
 [   0    1    3    0    0    0    0 1023    0    1]
 [   1    1    1    6    6    1    2    2  952    2]
 [   0    1    0    2    5    6    0    1    1  993]]
```

[ ]: