In [1]:
```
#Image classification using Deep Learning
import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.fashion_mnist.load_data()
```

In [3]:
```
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

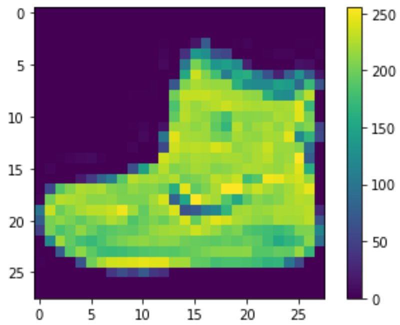
In [4]:
```

x_train.shape
```

Out[4]: (60000, 28, 28)

In [5]:
```

x_test.shape
```

Out[5]: (10000, 28, 28)

In [6]:
```
plt.imshow(x_train[0])
plt.colorbar()
plt.grid(False)
plt.show()
```



In [7]:
```


x_train[0].max()
```

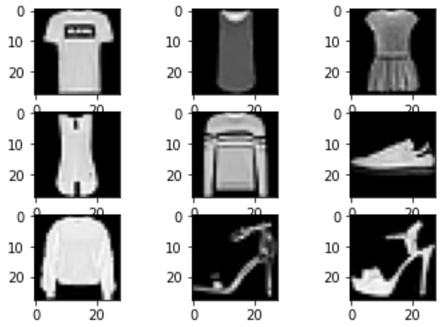
Out[7]: 255

In [8]:
```

x_test[0].max()
```

Out[8]: 255

In [9]:
```
# normalize values between 0 & 1

x_train = x_train/255
x_test = x_test/255

## another method to normalize

# x_train = tf.keras.utils.normalize(x_train, axis=1)
# x_test = tf.keras.utils.normalize(x_test, axis=1)
```

In [18]:
```
# check if images and class align

for i in range(1, 10):
    # Create a 3x3 grid and place the
    # image in ith position of grid
    plt.subplot(3, 3, i)
    # Insert ith image with the color map 'grap'
    plt.imshow(x_train[i], cmap=plt.get_cmap('gray'))

# Display the entire plot
plt.show()
```



In [11]:
```
model = tf.keras.models.Sequential(
[    tf.keras.layers.Flatten(input_shape=(28,28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')]
)

model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              optimizer='adam',
              metrics=['accuracy'])
```

In [12]:
```
model.fit(x_train, y_train, epochs=30, validation_split=0.2)
```

```
Epoch 1/30
1500/1500 [==============================] - 5s 3ms/step - loss: 0.5191 - accuracy: 0.8187 - val_loss: 0.4226 - val_accuracy: 0.8528
Epoch 2/30
1500/1500 [==============================] - 5s 3ms/step - loss: 0.3878 - accuracy: 0.8607 - val_loss: 0.3726 - val_accuracy: 0.8664
Epoch 3/30
1500/1500 [==============================] - 4s 2ms/step - loss: 0.3447 - accuracy: 0.8744 - val_loss: 0.3588 - val_accuracy: 0.8708
Epoch 4/30
1500/1500 [==============================] - 4s 3ms/step - loss: 0.3207 - accuracy: 0.8834 - val_loss: 0.3434 - val_accuracy: 0.8763
Epoch 5/30
1500/1500 [==============================] - 4s 3ms/step - loss: 0.3020 - accuracy: 0.8888 - val_loss: 0.3268 - val_accuracy: 0.8852
Epoch 6/30
1500/1500 [==============================] - 4s 3ms/step - loss: 0.2848 - accuracy: 0.8947 - val_loss: 0.3448 - val_accuracy: 0.8777
Epoch 7/30
1500/1500 [==============================] - 4s 2ms/step - loss: 0.2726 - accuracy: 0.8988 - val_loss: 0.3185 - val_accuracy: 0.8901
Epoch 8/30
1500/1500 [==============================] - 4s 3ms/step - loss: 0.2616 - accuracy: 0.9026 - val_loss: 0.3595 - val_accuracy: 0.8769
Epoch 9/30
1500/1500 [==============================] - 6s 4ms/step - loss: 0.2514 - accuracy: 0.9055 - val_loss: 0.3169 - val_accuracy: 0.8895
Epoch 10/30
1500/1500 [==============================] - 6s 4ms/step - loss: 0.2422 - accuracy: 0.9099 - val_loss: 0.3079 - val_accuracy: 0.8905
Epoch 11/30
1500/1500 [==============================] - 4s 3ms/step - loss: 0.2326 - accuracy: 0.9143 - val_loss: 0.3168 - val_accuracy: 0.8892
Epoch 12/30
1500/1500 [==============================] - 8s 5ms/step - loss: 0.2252 - accuracy: 0.9153 - val_loss: 0.3279 - val_accuracy: 0.8878
Epoch 13/30
1500/1500 [==============================] - 7s 4ms/step - loss: 0.2194 - accuracy: 0.9178 - val_loss: 0.3312 - val_accuracy: 0.8860
Epoch 14/30
1500/1500 [==============================] - 5s 3ms/step - loss: 0.2117 - accuracy: 0.9214 - val_loss: 0.3534 - val_accuracy: 0.8817
Epoch 15/30
1500/1500 [==============================] - 4s 3ms/step - loss: 0.2049 - accuracy: 0.9225 - val_loss: 0.3396 - val_accuracy: 0.8875
Epoch 16/30
1500/1500 [==============================] - 5s 4ms/step - loss: 0.2004 - accuracy: 0.9236 - val_loss: 0.3226 - val_accuracy: 0.8917
Epoch 17/30
1500/1500 [==============================] - 6s 4ms/step - loss: 0.1929 - accuracy: 0.9261 - val_loss: 0.3356 - val_accuracy: 0.8903
Epoch 18/30
1500/1500 [==============================] - 5s 4ms/step - loss: 0.1860 - accuracy: 0.9301 - val_loss: 0.3277 - val_accuracy: 0.8936
Epoch 19/30
1500/1500 [==============================] - 7s 5ms/step - loss: 0.1837 - accuracy: 0.9309 - val_loss: 0.3359 - val_accuracy: 0.8911
Epoch 20/30
1500/1500 [==============================] - 8s 5ms/step - loss: 0.1769 - accuracy: 0.9345 - val_loss: 0.3308 - val_accuracy: 0.8967
Epoch 21/30
1500/1500 [==============================] - 8s 5ms/step - loss: 0.1724 - accuracy: 0.9349 - val_loss: 0.3349 - val_accuracy: 0.8929
Epoch 22/30
1500/1500 [==============================] - 13s 9ms/step - loss: 0.1680 - accuracy: 0.9367 - val_loss: 0.3430 - val_accuracy: 0.8931
Epoch 23/30
1500/1500 [==============================] - 11s 7ms/step - loss: 0.1640 - accuracy: 0.9382 - val_loss: 0.3567 - val_accuracy: 0.8928
Epoch 24/30
1500/1500 [==============================] - 12s 8ms/step - loss: 0.1596 - accuracy: 0.9399 - val_loss: 0.3626 - val_accuracy: 0.8911
Epoch 25/30
1500/1500 [==============================] - 11s 7ms/step - loss: 0.1548 - accuracy: 0.9418 - val_loss: 0.3919 - val_accuracy: 0.8849
Epoch 26/30
1500/1500 [==============================] - 9s 6ms/step - loss: 0.1512 - accuracy: 0.9424 - val_loss: 0.3640 - val_accuracy: 0.8917
Epoch 27/30
1500/1500 [==============================] - 8s 5ms/step - loss: 0.1493 - accuracy: 0.9437 - val_loss: 0.3646 - val_accuracy: 0.8916
Epoch 28/30
1500/1500 [==============================] - 9s 6ms/step - loss: 0.1437 - accuracy: 0.9460 - val_loss: 0.3794 - val_accuracy: 0.8877
Epoch 29/30
1500/1500 [==============================] - 6s 4ms/step - loss: 0.1410 - accuracy: 0.9471 - val_loss: 0.3643 - val_accuracy: 0.8938
Epoch 30/30
1500/1500 [==============================] - 6s 4ms/step - loss: 0.1379 - accuracy: 0.9476 - val_loss: 0.3927 - val_accuracy: 0.8932
```

Out[12]:
```
<keras.callbacks.History at 0x2b19b59d5e0>
```

In [43]:
```
#overfitting problems
from tensorflow.keras.optimizers import Adam
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=3)

model = tf.keras.models.Sequential(
    [tf.keras.layers.Flatten(input_shape=(28,28)),
     tf.keras.layers.Dense(128, activation='relu'),
     tf.keras.layers.Dropout(0.25),
     tf.keras.layers.Dense(10, activation='softmax')]
)


model.compile(optimizer=Adam(learning_rate=1e-3),
              loss='sparse_categorical_crossentropy',
              metrics=['sparse_categorical_accuracy'])

history = model.fit(x_train, y_train, epochs=20, validation_split=0.25, callbacks=[early_stopping])

#ReLu function: it's a linear activation function.
#It is less susceptible to vanishing gradients that prevent deep models from being trained, although it can suffer from other problems
#like saturated or "dead" units. It is calculated as  max(0.0,x) , where if the input value (x) is negative, then a value 0.0 is
#returned, otherwise, the value is returned.
```

```
Epoch 1/20
1407/1407 [==============================] - 4s 3ms/step - loss: 0.5707 - sparse_categorical_accuracy: 0.7984 - val_loss: 0.4192 - val_spa
rse_categorical_accuracy: 0.8455
Epoch 2/20
1407/1407 [==============================] - 4s 3ms/step - loss: 0.4274 - sparse_categorical_accuracy: 0.8446 - val_loss: 0.3815 - val_spa
rse_categorical_accuracy: 0.8621
Epoch 3/20
1407/1407 [==============================] - 4s 3ms/step - loss: 0.3855 - sparse_categorical_accuracy: 0.8588 - val_loss: 0.3679 - val_spa
rse_categorical_accuracy: 0.8652
Epoch 4/20
1407/1407 [==============================] - 3s 2ms/step - loss: 0.3670 - sparse_categorical_accuracy: 0.8656 - val_loss: 0.3548 - val_spa
rse_categorical_accuracy: 0.8692
Epoch 5/20
1407/1407 [==============================] - 3s 2ms/step - loss: 0.3469 - sparse_categorical_accuracy: 0.8710 - val_loss: 0.3869 - val_spa
rse_categorical_accuracy: 0.8516
Epoch 6/20
1407/1407 [==============================] - 4s 3ms/step - loss: 0.3354 - sparse_categorical_accuracy: 0.8760 - val_loss: 0.3324 - val_spa
rse_categorical_accuracy: 0.8799
Epoch 7/20
1407/1407 [==============================] - 4s 3ms/step - loss: 0.3243 - sparse_categorical_accuracy: 0.8808 - val_loss: 0.3424 - val_spa
rse_categorical_accuracy: 0.8711
Epoch 8/20
1407/1407 [==============================] - 3s 2ms/step - loss: 0.3192 - sparse_categorical_accuracy: 0.8820 - val_loss: 0.3272 - val_spa
rse_categorical_accuracy: 0.8791
Epoch 9/20
1407/1407 [==============================] - 4s 3ms/step - loss: 0.3037 - sparse_categorical_accuracy: 0.8874 - val_loss: 0.3353 - val_spa
rse_categorical_accuracy: 0.8783
Epoch 10/20
1407/1407 [==============================] - 4s 3ms/step - loss: 0.2995 - sparse_categorical_accuracy: 0.8884 - val_loss: 0.3255 - val_spa
rse_categorical_accuracy: 0.8823
Epoch 11/20
1407/1407 [==============================] - 4s 3ms/step - loss: 0.2906 - sparse_categorical_accuracy: 0.8916 - val_loss: 0.3180 - val_spa
rse_categorical_accuracy: 0.8838
Epoch 12/20
1407/1407 [==============================] - 4s 3ms/step - loss: 0.2827 - sparse_categorical_accuracy: 0.8952 - val_loss: 0.3165 - val_spa
rse_categorical_accuracy: 0.8876
Epoch 13/20
1407/1407 [==============================] - 4s 3ms/step - loss: 0.2800 - sparse_categorical_accuracy: 0.8930 - val_loss: 0.3322 - val_spa
rse_categorical_accuracy: 0.8793
Epoch 14/20
1407/1407 [==============================] - 3s 2ms/step - loss: 0.2741 - sparse_categorical_accuracy: 0.8975 - val_loss: 0.3262 - val_spa
rse_categorical_accuracy: 0.8845
Epoch 15/20
1407/1407 [==============================] - 4s 3ms/step - loss: 0.2666 - sparse_categorical_accuracy: 0.9001 - val_loss: 0.3172 - val_spa
rse_categorical_accuracy: 0.8879
```

In [44]:
```
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_5 (Flatten)         (None, 784)               0

 dense_10 (Dense)            (None, 128)               100480

 dropout_4 (Dropout)         (None, 128)               0

 dense_11 (Dense)            (None, 10)                1290

=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```

In [45]:
```
test_loss, test_acc = model.evaluate(x_train, y_train)

print('Test accuracy:', test_acc)
```

```
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2470 - sparse_categorical_accuracy: 0.9098
Test accuracy: 0.9098166823387146
```

In [46]:
```
predictions = model.predict(x_test)
predictions[0]
```

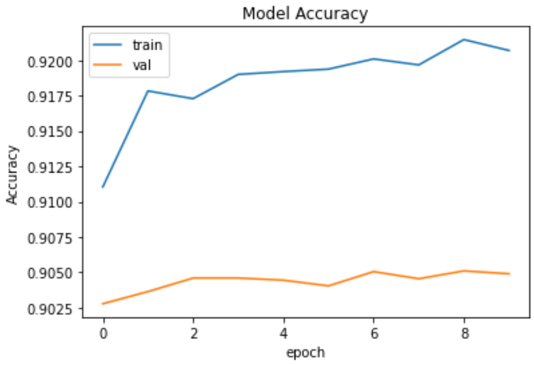
```
313/313 [==============================] - 0s 985us/step
```

Out[46]:
```
array([9.5804626e-07, 5.4533555e-09, 4.0153523e-08, 3.5564550e-08,
       6.7747530e-10, 1.1437546e-02, 1.4985049e-07, 2.9816655e-03,
       8.8535188e-07, 9.8557866e-01], dtype=float32)
```

In [47]:
```
history = model.fit(
    x_train.astype(np.float32), y_train.astype(np.float32),
    epochs=10,
    steps_per_epoch=100,
    validation_split=0.33
)
```

```
Epoch 1/10
100/100 [==============================] - 2s 13ms/step - loss: 0.2391 - sparse_categorical_accuracy: 0.9110 - val_loss: 0.2731 - val_spar
se_categorical_accuracy: 0.9028
Epoch 2/10
100/100 [==============================] - 1s 14ms/step - loss: 0.2208 - sparse_categorical_accuracy: 0.9178 - val_loss: 0.2713 - val_spar
se_categorical_accuracy: 0.9036
Epoch 3/10
100/100 [==============================] - 1s 12ms/step - loss: 0.2213 - sparse_categorical_accuracy: 0.9173 - val_loss: 0.2716 - val_spar
se_categorical_accuracy: 0.9046
Epoch 4/10
100/100 [==============================] - 2s 18ms/step - loss: 0.2183 - sparse_categorical_accuracy: 0.9190 - val_loss: 0.2716 - val_spar
se_categorical_accuracy: 0.9046
Epoch 5/10
100/100 [==============================] - 1s 14ms/step - loss: 0.2186 - sparse_categorical_accuracy: 0.9192 - val_loss: 0.2702 - val_spar
se_categorical_accuracy: 0.9044
Epoch 6/10
100/100 [==============================] - 1s 13ms/step - loss: 0.2151 - sparse_categorical_accuracy: 0.9194 - val_loss: 0.2730 - val_spar
se_categorical_accuracy: 0.9040
Epoch 7/10
100/100 [==============================] - 2s 15ms/step - loss: 0.2135 - sparse_categorical_accuracy: 0.9201 - val_loss: 0.2704 - val_spar
se_categorical_accuracy: 0.9051
Epoch 8/10
100/100 [==============================] - 1s 13ms/step - loss: 0.2161 - sparse_categorical_accuracy: 0.9197 - val_loss: 0.2716 - val_spar
se_categorical_accuracy: 0.9046
Epoch 9/10
100/100 [==============================] - 1s 12ms/step - loss: 0.2116 - sparse_categorical_accuracy: 0.9215 - val_loss: 0.2720 - val_spar
se_categorical_accuracy: 0.9051
Epoch 10/10
100/100 [==============================] - 1s 13ms/step - loss: 0.2103 - sparse_categorical_accuracy: 0.9207 - val_loss: 0.2719 - val_spar
se_categorical_accuracy: 0.9049
```

In [50]:
```
plt.plot(history.history['sparse_categorical_accuracy'])
plt.plot(history.history['val_sparse_categorical_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



In [ ]:

In [ ]: