In [33]:
```
# Use Conv2D and Max pooling on Mnist Fashion dataset to predict the class
```

In [34]:
```
import numpy as np
import pandas as pd
import keras
import matplotlib.pyplot as plt
```

In [34]:
```

```

In [35]:
```
from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D,Activation,Flatten,Dense,Dropout
from keras.datasets import fashion_mnist
```

In [36]:
```
(train_X,train_Y), (test_X,test_Y) = fashion_mnist.load_data()
```

In [37]:
```
# analysisng the data
```

In [38]:
```
import numpy as np
from keras.utils import to_categorical
import matplotlib.pyplot as plt
%matplotlib inline

print('Training data shape : ', train_X.shape, train_Y.shape)

print('Testing data shape : ', test_X.shape, test_Y.shape)
```

```
Training data shape :  (60000, 28, 28) (60000,)
Testing data shape :  (10000, 28, 28) (10000,)
```

In [39]:
```
# Find the unique numbers from the train labels
classes = np.unique(train_Y)
nClasses = len(classes)
print('Total number of outputs : ', nClasses)
print('Output classes : ', classes)
```

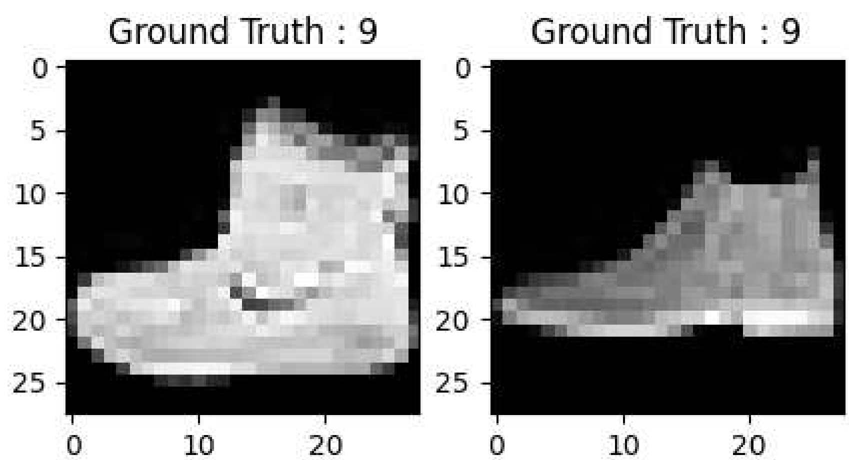
```
Total number of outputs :  10
Output classes :  [0 1 2 3 4 5 6 7 8 9]
```

In [40]:
```
plt.figure(figsize=[5,5])

# Display the first image in training data
plt.subplot(121)
plt.imshow(train_X[0,:,:], cmap='gray')
plt.title("Ground Truth : {}".format(train_Y[0]))

# Display the first image in testing data
plt.subplot(122)
plt.imshow(test_X[0,:,:], cmap='gray')
plt.title("Ground Truth : {}".format(test_Y[0]))
```

Out[40]:
```
Text(0.5, 1.0, 'Ground Truth : 9')
```



In [41]:
```
train_X = train_X.reshape(-1, 28,28, 1)
test_X = test_X.reshape(-1, 28,28, 1)
train_X.shape, test_X.shape
```

Out[41]:
```
((60000, 28, 28, 1), (10000, 28, 28, 1))
```

In [41]:
```

```

In [42]:
```
train_X = train_X.astype('float32')
test_X = test_X.astype('float32')
train_X = train_X / 255.
test_X = test_X / 255.
```

In [43]:
```
# using one hot encoding on y train labels
```

In [44]:
```
# Change the labels from categorical to one-hot encoding
train_Y_one_hot = to_categorical(train_Y)
test_Y_one_hot = to_categorical(test_Y)

# Display the change for category label using one-hot encoding
print('Original label:', train_Y[0])
print('After conversion to one-hot:', train_Y_one_hot[0])
```

```
Original label: 9
After conversion to one-hot: [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
```

In [45]:
```
from sklearn.model_selection import train_test_split
train_X,valid_X,train_label,valid_label = train_test_split(train_X, train_Y_on
```

In [45]:
```
```

In [32]:
```
batch_size = 64
epochs = 5
num_classes = 10
```

In [46]:
```
fashion_model = Sequential()
fashion_model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shap
fashion_model.add(MaxPooling2D(pool_size=(2, 2)))
fashion_model.add(Conv2D(64, (3, 3), activation='relu'))
fashion_model.add(MaxPooling2D(pool_size=(2, 2)))
fashion_model.add(Flatten())
fashion_model.add(Dense(128, activation='relu'))
fashion_model.add(Dropout(0.5))
fashion_model.add(Dense(num_classes, activation='softmax'))
```

In [47]:
```
fashion_model.compile(loss=keras.losses.categorical_crossentropy, optimizer=ke
```

In [48]:
```
fashion_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_2 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_2 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 conv2d_3 (Conv2D)           (None, 11, 11, 64)        18496

 max_pooling2d_3 (MaxPooling  (None, 5, 5, 64)         0
 2D)

 flatten_1 (Flatten)         (None, 1600)              0

 dense_2 (Dense)             (None, 128)               204928

 dropout_1 (Dropout)         (None, 128)               0

 dense_3 (Dense)             (None, 10)                1290

=================================================================
Total params: 225,034
Trainable params: 225,034
Non-trainable params: 0
_________________________________________________________________
```

In [31]:
```
fashion_train = fashion_model.fit(train_X, train_label, batch_size=batch_size,
```

```
Epoch 1/20
750/750 [==============================] - 49s 65ms/step - loss: 0.1569 - acc
uracy: 0.9389 - val_loss: 0.2814 - val_accuracy: 0.9114
Epoch 2/20
750/750 [==============================] - 52s 69ms/step - loss: 0.1503 - acc
uracy: 0.9418 - val_loss: 0.2886 - val_accuracy: 0.9124
Epoch 3/20
750/750 [==============================] - 50s 66ms/step - loss: 0.1467 - acc
uracy: 0.9429 - val_loss: 0.2888 - val_accuracy: 0.9152
Epoch 4/20
750/750 [==============================] - 49s 66ms/step - loss: 0.1413 - acc
uracy: 0.9444 - val_loss: 0.2970 - val_accuracy: 0.9158
Epoch 5/20
750/750 [==============================] - 47s 63ms/step - loss: 0.1380 - acc
uracy: 0.9468 - val_loss: 0.2875 - val_accuracy: 0.9168
Epoch 6/20
750/750 [==============================] - 51s 68ms/step - loss: 0.1343 - acc
uracy: 0.9479 - val_loss: 0.3094 - val_accuracy: 0.9123
Epoch 7/20
750/750 [==============================] - 51s 68ms/step - loss: 0.1313 - acc
uracy: 0.9486 - val_loss: 0.2948 - val_accuracy: 0.9152
Epoch 8/20
750/750 [==============================] - 49s 65ms/step - loss: 0.1246 - acc
uracy: 0.9505 - val_loss: 0.3099 - val_accuracy: 0.9154
Epoch 9/20
750/750 [==============================] - 50s 66ms/step - loss: 0.1216 - acc
uracy: 0.9523 - val_loss: 0.2996 - val_accuracy: 0.9181
Epoch 10/20
750/750 [==============================] - 50s 66ms/step - loss: 0.1213 - acc
uracy: 0.9517 - val_loss: 0.3214 - val_accuracy: 0.9137
Epoch 11/20
750/750 [==============================] - 48s 64ms/step - loss: 0.1161 - acc
uracy: 0.9540 - val_loss: 0.3206 - val_accuracy: 0.9154
Epoch 12/20
750/750 [==============================] - 51s 68ms/step - loss: 0.1141 - acc
uracy: 0.9540 - val_loss: 0.3299 - val_accuracy: 0.9136
Epoch 13/20
750/750 [==============================] - 51s 68ms/step - loss: 0.1112 - acc
uracy: 0.9546 - val_loss: 0.3257 - val_accuracy: 0.9149
Epoch 14/20
750/750 [==============================] - 49s 65ms/step - loss: 0.1070 - acc
uracy: 0.9561 - val_loss: 0.3371 - val_accuracy: 0.9147
Epoch 15/20
750/750 [==============================] - 47s 63ms/step - loss: 0.1049 - acc
uracy: 0.9577 - val_loss: 0.3556 - val_accuracy: 0.9126
Epoch 16/20
750/750 [==============================] - 49s 65ms/step - loss: 0.1032 - acc
uracy: 0.9586 - val_loss: 0.3597 - val_accuracy: 0.9148
Epoch 17/20
750/750 [==============================] - 49s 65ms/step - loss: 0.0997 - acc
uracy: 0.9604 - val_loss: 0.3666 - val_accuracy: 0.9158
Epoch 18/20
750/750 [==============================] - 47s 63ms/step - loss: 0.0994 - acc
uracy: 0.9599 - val_loss: 0.3808 - val_accuracy: 0.9158
Epoch 19/20
750/750 [==============================] - 51s 68ms/step - loss: 0.1007 - acc
uracy: 0.9589 - val_loss: 0.3737 - val_accuracy: 0.9142
```

```
Epoch 20/20
750/750 [==============================] - 51s 68ms/step - loss: 0.0956 - acc
uracy: 0.9610 - val_loss: 0.3670 - val_accuracy: 0.9139
```

In [49]:
```
predictions=fashion_model.predict(test_X)
```

```
313/313 [==============================] - 3s 10ms/step
```

In [50]:
```
# evaluate the model
```

In [51]:
```
test_eval = fashion_model.evaluate(test_X, test_Y_one_hot, verbose=0)
```

In [52]:
```
print('Test loss:', test_eval[0])
print('Test accuracy:', test_eval[1])
```

```
Test loss: 2.3254783153533936
Test accuracy: 0.03370000049471855
```

In [53]:
```
('Test loss:', 0.46366268818555401)
('Test accuracy:', 0.91839999999999999)
```

Out[53]:
```
('Test accuracy:', 0.9184)
```

In [53]:

In [53]: