In [1]:

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
```

Set training parameters

In [2]:

```
# Parameters
learning_rate = 0.001
training_epochs = 20
batch_size = 32
```

Load Fashion_MNIST data Set train & test data

In [3]:

```
# Load and prepare the MNIST dataset
mnist = keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Split dataset
x_train = x_train.reshape(x_train.shape[0], 28, 28, 1).astype("float32") / 255.0
x_test = x_test.reshape(x_test.shape[0], 28, 28, 1).astype("float32") / 255.0

# Use tf.data to batch and shuffle the dataset
train_data = tf.data.Dataset.from_tensor_slices((x_train, y_train)).shuffle(10000).batch(batch_size)
test_data = tf.data.Dataset.from_tensor_slices((x_test, y_test)).batch(batch_size)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/tr
ain-labels-idx1-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/train-labels-idx1-ubyte.gz)
32768/29515 [================================] - 0s 0us/step
40960/29515 [========================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/tr
ain-images-idx3-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/train-images-idx3-ubyte.gz)
26427392/26421880 [==============================] - 0s 0us/step
26435584/26421880 [==============================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t1
0k-labels-idx1-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/t10k-labels-idx1-ubyte.gz)
16384/5148 [===============================================================================
======================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t1
0k-images-idx3-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/t10k-images-idx3-ubyte.gz)
4423680/4422102 [==============================] - 0s 0us/step
4431872/4422102 [==============================] - 0s 0us/step
```

In [4]:

```
dim_x = np.shape(x_train)[1:]
```

Set model

- **Structure**: Conv2D + MaxPool2D + Dense + Dense
- **Loss function**: Cross-entropy
- **Optimization method**: Adam

In [11]:

```
class MLP(tf.keras.Model):
  """ A basic cnn class for tensorflow
  Extends:
    tf.keras.Model
  """
  def __init__(self, **kwargs):
    super(MLP, self).__init__()
    self.__dict__.update(kwargs)

    # Set network
    self.nn = keras.Sequential(self.nn_desc)

    # Set loss
    self.loss_fn = keras.losses.SparseCategoricalCrossentropy()

  def __call__(self, x_in):
    return self.nn(x_in)

  @tf.function
  def compute_loss(self, y_true, y_pred):
    return self.loss_fn(y_true, y_pred)

  def compute_gradients(self, x_in, y_in):
    with tf.GradientTape() as tape:
      y_pred = self.nn(x_in)
      loss_out = self.compute_loss(y_in, y_pred)

    cg_out = tape.gradient(loss_out, self.trainable_variables)
    return cg_out, y_pred, loss_out

  @tf.function
  def train(self, x_in, y_in):
    """ Trains model. """
    gradients, y_pred, loss_out = self.compute_gradients(x_in, y_in)
    self.optimizer.apply_gradients(zip(gradients, self.trainable_variables))
    return y_pred, loss_out
```

In [19]:

```
# nn_desc = [keras.layers.InputLayer(input_shape=dim_x, dtype=tf.float32),
#            keras.layers.Dense(units = 256,activation = 'relu'),
#         #    keras.layers.Conv2D(filters=32, kernel_size=5, strides=(1, 1), activation="relu"),
#            keras.layers.MaxPool2D(pool_size=(2, 2), strides=(2, 2), padding='valid'),
#         #    keras.layers.Conv2D(filters=32, kernel_size=5, strides=(1, 1), activation="relu"),
#            keras.layers.MaxPool2D(pool_size=(2, 2), strides=(2, 2), padding='valid'),
#            keras.layers.Flatten(),
#            keras.layers.Dense(units=128, activation='relu'),
#            keras.layers.Dense(units=10, activation='softmax')]
model =[
    keras.layers.InputLayer(input_shape=dim_x, dtype=tf.float32),
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(512, activation='relu'),
    keras.layers.Dense(256,activation='relu'),
    keras.layers.Dense(10, activation='softmax')
]

# Set optimizer
optimizer = tf.optimizers.Adam(learning_rate)
```

In [20]:

```
cnn = MLP(learning_rate=learning_rate, nn_desc=nn_desc, optimizer=optimizer)
```

###Train model

In [21]:

```
# Select metrics to measure the loss and the accuracy of the model
train_loss = keras.metrics.Mean(name='train_loss')
train_accuracy = keras.metrics.SparseCategoricalAccuracy(name='train_accuracy')

test_loss = keras.metrics.Mean(name='test_loss')
test_accuracy = keras.metrics.SparseCategoricalAccuracy(name='test_accuracy')
```

In [22]:

```
# Run training for the given number of steps
for epoch in range(training_epochs):
  for x_batch, y_batch in train_data:
    y_pred, loss = cnn.train(x_batch, y_batch)
    train_loss(loss)
    train_accuracy(y_batch, y_pred)


  for x_batch_t, y_batch_t in test_data:
    y_pred_t = cnn(x_batch_t)
    loss_t = cnn.compute_loss(y_batch_t, y_pred_t)
    test_loss(loss_t)
    test_accuracy(y_batch_t, y_pred_t)

  template = 'Epoch {:d}, Loss: {:f}, Accuracy: {:f}, Test Loss: {:f}, Test Accuracy: {:f}'
  print(template.format(epoch+1,
                        train_loss.result(),
                        train_accuracy.result()*100,
                        test_loss.result(),
                        test_accuracy.result()*100))

  # Reset the metrics for the next epoch
  train_loss.reset_states()
  train_accuracy.reset_states()
  test_loss.reset_states()
  test_accuracy.reset_states()
```

```
Epoch 1, Loss: 0.428881, Accuracy: 83.693329, Test Loss: 0.523508, Test Accuracy: 8
0.709999
Epoch 2, Loss: 0.424860, Accuracy: 83.836670, Test Loss: 0.512764, Test Accuracy: 8
0.940002
Epoch 3, Loss: 0.421147, Accuracy: 84.001663, Test Loss: 0.503147, Test Accuracy: 8
1.190002
Epoch 4, Loss: 0.416237, Accuracy: 84.135002, Test Loss: 0.508038, Test Accuracy: 8
0.769997
Epoch 5, Loss: 0.413824, Accuracy: 84.246666, Test Loss: 0.500144, Test Accuracy: 8
1.370003
Epoch 6, Loss: 0.410449, Accuracy: 84.358337, Test Loss: 0.505552, Test Accuracy: 8
1.419998
Epoch 7, Loss: 0.406464, Accuracy: 84.513329, Test Loss: 0.518550, Test Accuracy: 8
0.269997
Epoch 8, Loss: 0.404183, Accuracy: 84.658333, Test Loss: 0.512587, Test Accuracy: 8
1.019997
Epoch 9, Loss: 0.401061, Accuracy: 84.751663, Test Loss: 0.497955, Test Accuracy: 8
1.650002
Epoch 10, Loss: 0.397658, Accuracy: 84.856667, Test Loss: 0.495950, Test Accuracy: 8
2.000000
Epoch 11, Loss: 0.396219, Accuracy: 84.991661, Test Loss: 0.507836, Test Accuracy: 8
1.209999
Epoch 12, Loss: 0.392647, Accuracy: 84.995003, Test Loss: 0.506959, Test Accuracy: 8
1.070000
Epoch 13, Loss: 0.390075, Accuracy: 85.119995, Test Loss: 0.536122, Test Accuracy: 8
0.220001
Epoch 14, Loss: 0.386038, Accuracy: 85.343330, Test Loss: 0.506394, Test Accuracy: 8
1.370003
Epoch 15, Loss: 0.385565, Accuracy: 85.175003, Test Loss: 0.497841, Test Accuracy: 8
1.610001
Epoch 16, Loss: 0.382509, Accuracy: 85.533333, Test Loss: 0.513609, Test Accuracy: 8
```

```
1.230003
Epoch 17, Loss: 0.380938, Accuracy: 85.495003, Test Loss: 0.512330, Test Accuracy: 8
1.230003
Epoch 18, Loss: 0.379070, Accuracy: 85.493332, Test Loss: 0.502122, Test Accuracy: 8
1.239998
Epoch 19, Loss: 0.376794, Accuracy: 85.541664, Test Loss: 0.518050, Test Accuracy: 8
1.349998
Epoch 20, Loss: 0.373956, Accuracy: 85.691666, Test Loss: 0.527229, Test Accuracy: 8
1.419998
```

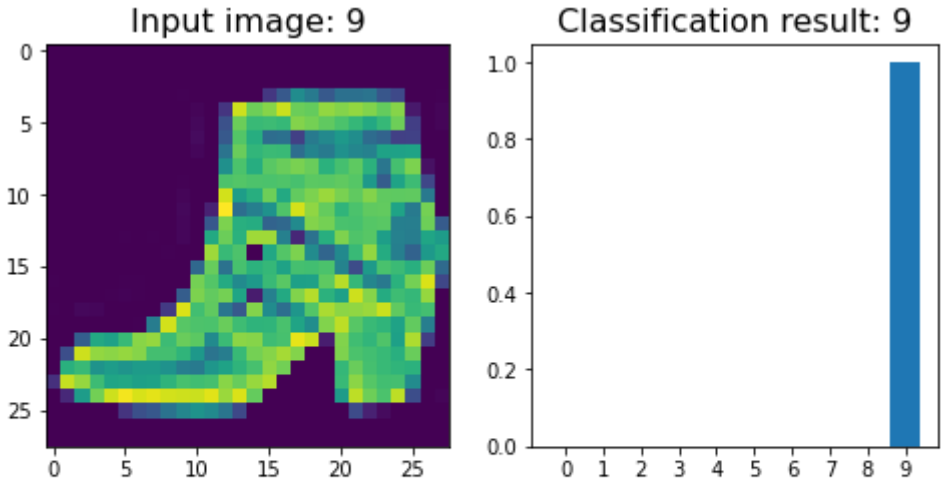
###**Test model** Test trained CNN model

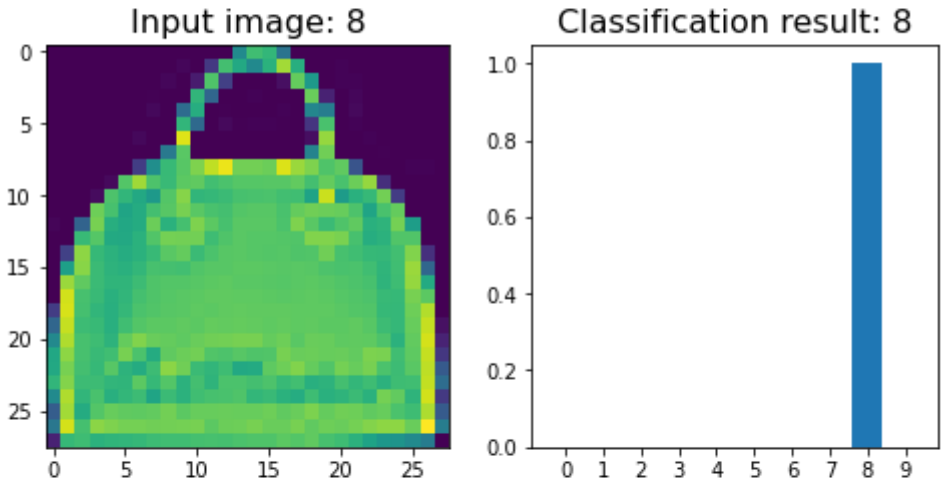
In [23]:

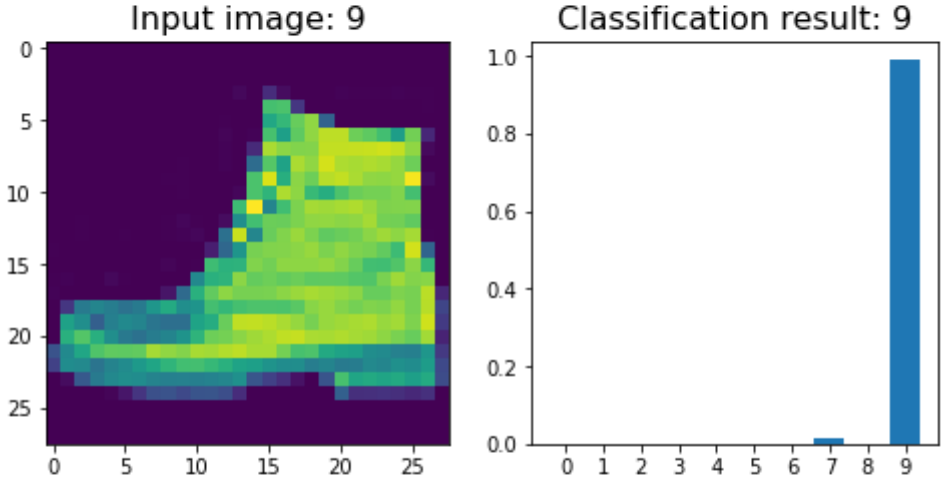
```
# Select test data
num_test_sel = 4
num_test = x_test.shape[0]
idx_rand = np.random.permutation(np.arange(0, num_test))
idx_rand_sel = idx_rand[0:num_test_sel]

x_sel = x_test[idx_rand_sel, :, :, :]
y_sel = y_test[idx_rand_sel]

y_out = cnn(x_sel)
y_out = y_out.numpy()
y_sel_pred = np.argmax(y_out, axis=1)
```

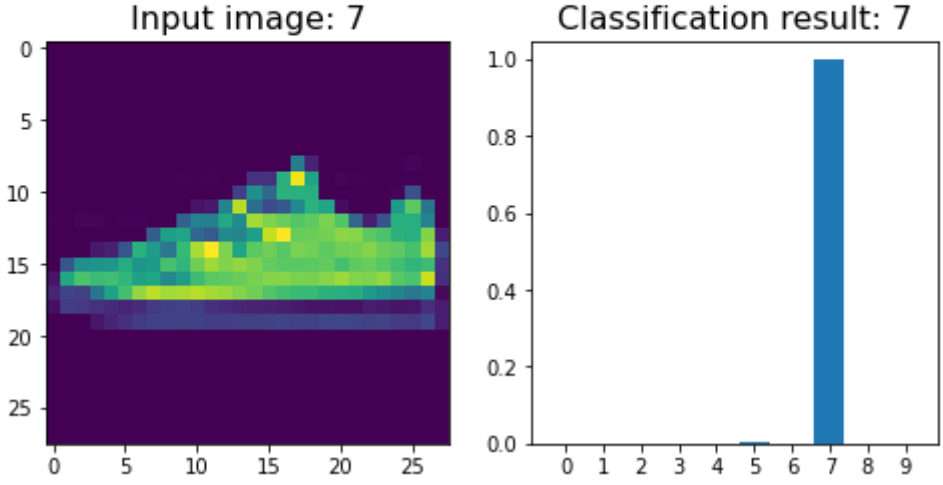
In [24]:

```
# Plot test result
import matplotlib.pyplot as plt
from matplotlib import gridspec
barx = np.arange(0,10)

fig = plt.figure(figsize=(8, 4.25 * num_test_sel))
gs = gridspec.GridSpec(nrows=num_test_sel,
                       ncols=2,
                       height_ratios=[1]*num_test_sel,
                       width_ratios=[1, 1]
                       )

for nidx_d in range(0, num_test_sel):
  _x_sel_plot = x_sel[nidx_d, :]
  x_sel_plot = np.reshape(_x_sel_plot, (28, 28))
  y_sel_plot = y_sel[nidx_d]
  y_out_plot = y_out[nidx_d, :]
  y_sel_pred_plot = y_sel_pred[nidx_d]
  ax0 = plt.subplot(gs[nidx_d, 0])
  ax0.imshow(x_sel_plot)
  ax0.title.set_text('Input image: {:d}'.format(y_sel_plot))
  ax0.title.set_fontsize(16)

  ax1 = plt.subplot(gs[nidx_d, 1])
  ax1.bar(barx, y_out_plot.reshape(-1))
  plt.xticks(barx)
  ax1.title.set_text('Classification result: {:d}'.format(y_sel_pred_plot))
  ax1.title.set_fontsize(16)

plt.show()
```


Input image: 9
Classification result: 9


Input image: 8
Classification result: 8


Input image: 9
Classification result: 9


Input image: 7
Classification result: 7

In [ ]:

```
```