In [107]:

```
import keras
from keras.datasets import mnist
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.utils.vis_utils import plot_model
from keras_tqdm import TQDMNotebookCallback
import matplotlib.pyplot as plt

import tensorflow as tf
```

In [108]:

```
# Loading the data
import tensorflow as tf
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

In [109]:

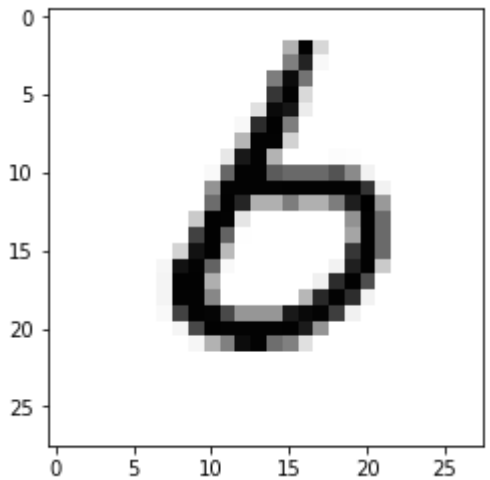
```
import matplotlib.pyplot as plt
%matplotlib inline
# You may select anything up to 60,000
image_index = 5998

print(y_train[image_index])
plt.imshow(x_train[image_index], cmap='Greys')
```

```
6
```

Out[109]:

```
<matplotlib.image.AxesImage at 0x1a6537fb70>
```



In [110]:

```
#reshaping the input data
x_train.shape
```

Out[110]:

```
(60000, 28, 28)
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

In [111]:

```
# input image dimensions
img_rows, img_cols = 28, 28
batch_size = 128
num_classes = 10
epochs = 15
# the data, split between train and test sets
if K.image_data_format() == 'channels_first':
    x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
    x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
    input_shape = (1, img_rows, img_cols)
else:
    x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
    x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
    input_shape = (img_rows, img_cols, 1)

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print('x_train shape:', x_train.shape)
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
```

In [112]:

```
# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

In [113]:

```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape)
model.add(Conv2D(64, (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(0.5))
model.add(Dense(num_classes, activation='softmax'))

model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adadelta(),
              metrics=['accuracy'])
```

In [ ]:

```

```

In [114]:

```
 # start train
history = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=20,
          verbose=0,
          validation_data=(x_test, y_test), callbacks=[TQDMNotebookCallback()])
```

Training 100% 20/20 [1:04:04<00:00, 167.08s/it]

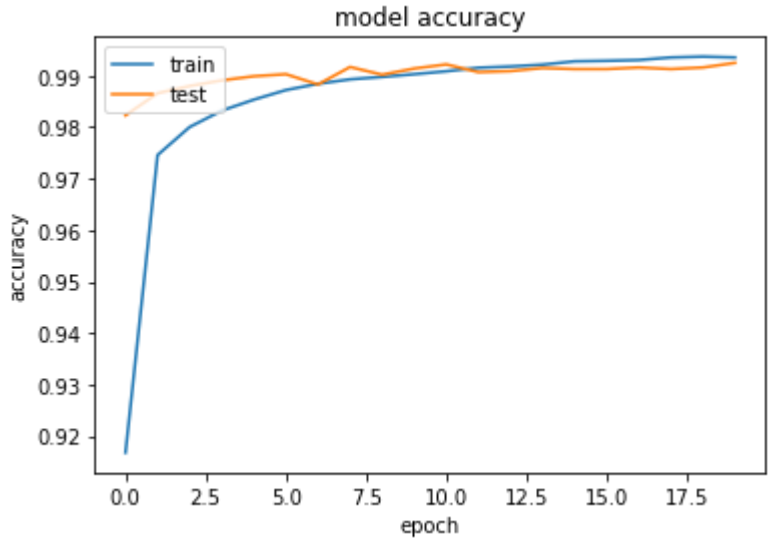
In [115]:

```
 # save model
history.history['acc']
# evaluate and print test accuracy
score = model.evaluate(x_test, y_test, verbose=0)
print("Accuracy on test set: ",score[1])
```

```
Accuracy on test set:  0.9925
```

In [116]:

```
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



In [ ]: