In [9]:

```python
from __future__ import print_function
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.layers.normalization import BatchNormalization

batch_size = 128
num_classes = 10
epochs = 12

# input image dimensions
img_rows, img_cols = 28, 28

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

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')
```

```
# 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)
```

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

CNN model using 3*3 Kernel

In [0]:

```
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'])

history = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:66: The name tf.get_default_graph is deprec
ated. Please use tf.compat.v1.get_default_graph instead.
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:541: The name tf.placeholder is deprecated.
Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:4432: The name tf.random_uniform is depreca
ted. Please use tf.random.uniform instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:4267: The name tf.nn.max_pool is deprecate
d. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:148: The name tf.placeholder_with_default i
s deprecated. Please use tf.compat.v1.placeholder_with_default instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:3733: calling dropout (from tensorflow.pyth
on.ops.nn_ops) with keep_prob is deprecated and will be removed in a fu
ture version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate =
1 - keep_prob`.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/op
timizers.py:793: The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:3576: The name tf.log is deprecated. Please
use tf.math.log instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorfl
ow_core/python/ops/math_grad.py:1424: where (from tensorflow.python.op
s.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
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:1033: The name tf.assign_add is deprecated.
```

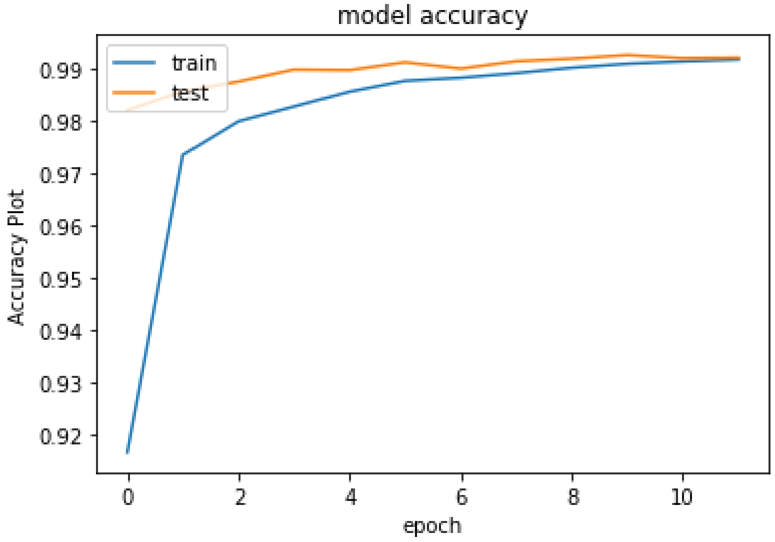
```
Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:1020: The name tf.assign is deprecated. Ple
ase use tf.compat.v1.assign instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:3005: The name tf.Session is deprecated. Pl
ease use tf.compat.v1.Session instead.

Train on 60000 samples, validate on 10000 samples
Epoch 1/12
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:190: The name tf.get_default_session is dep
recated. Please use tf.compat.v1.get_default_session instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:197: The name tf.ConfigProto is deprecated.
Please use tf.compat.v1.ConfigProto instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:207: The name tf.global_variables is deprec
ated. Please use tf.compat.v1.global_variables instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:216: The name tf.is_variable_initialized is
deprecated. Please use tf.compat.v1.is_variable_initialized instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:223: The name tf.variables_initializer is d
eprecated. Please use tf.compat.v1.variables_initializer instead.

60000/60000 [==============================] - 145s 2ms/step - loss: 0.
2696 - acc: 0.9166 - val_loss: 0.0559 - val_acc: 0.9820
Epoch 2/12
60000/60000 [==============================] - 143s 2ms/step - loss: 0.
0873 - acc: 0.9734 - val_loss: 0.0454 - val_acc: 0.9856
Epoch 3/12
60000/60000 [==============================] - 143s 2ms/step - loss: 0.
```

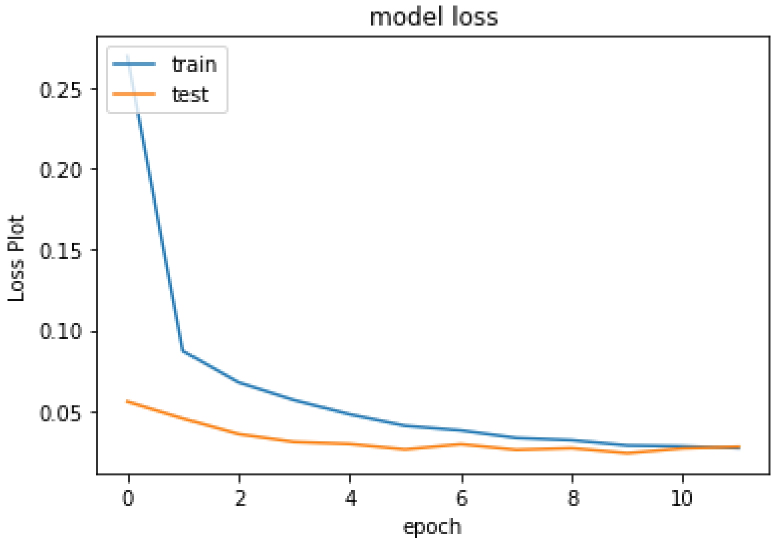
```
0678 - acc: 0.9799 - val_loss: 0.0357 - val_acc: 0.9875
Epoch 4/12
60000/60000 [==============================] - 143s 2ms/step - loss: 0.
0568 - acc: 0.9827 - val_loss: 0.0308 - val_acc: 0.9898
Epoch 5/12
60000/60000 [==============================] - 143s 2ms/step - loss: 0.
0480 - acc: 0.9855 - val_loss: 0.0296 - val_acc: 0.9897
Epoch 6/12
60000/60000 [==============================] - 143s 2ms/step - loss: 0.
0409 - acc: 0.9876 - val_loss: 0.0262 - val_acc: 0.9912
Epoch 7/12
60000/60000 [==============================] - 143s 2ms/step - loss: 0.
0380 - acc: 0.9882 - val_loss: 0.0294 - val_acc: 0.9900
Epoch 8/12
60000/60000 [==============================] - 144s 2ms/step - loss: 0.
0333 - acc: 0.9891 - val_loss: 0.0259 - val_acc: 0.9914
Epoch 9/12
60000/60000 [==============================] - 142s 2ms/step - loss: 0.
0319 - acc: 0.9901 - val_loss: 0.0269 - val_acc: 0.9919
Epoch 10/12
60000/60000 [==============================] - 143s 2ms/step - loss: 0.
0286 - acc: 0.9909 - val_loss: 0.0238 - val_acc: 0.9926
Epoch 11/12
60000/60000 [==============================] - 144s 2ms/step - loss: 0.
0281 - acc: 0.9914 - val_loss: 0.0268 - val_acc: 0.9920
Epoch 12/12
60000/60000 [==============================] - 144s 2ms/step - loss: 0.
0270 - acc: 0.9917 - val_loss: 0.0279 - val_acc: 0.9921
Test loss: 0.02786154808707215
Test accuracy: 0.9921
```

In [0]:

```
import matplotlib.pyplot as plt

print(model.metrics_names)
print(score)
print(history.history.keys())
#  Accuracy of the model
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('Accuracy Plot')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# Loss of the model
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('Loss Plot')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

```
['loss', 'acc']
[0.027861548087072l5, 0.9921]
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```





CNN model using 2*2 Kernel

In [0]:

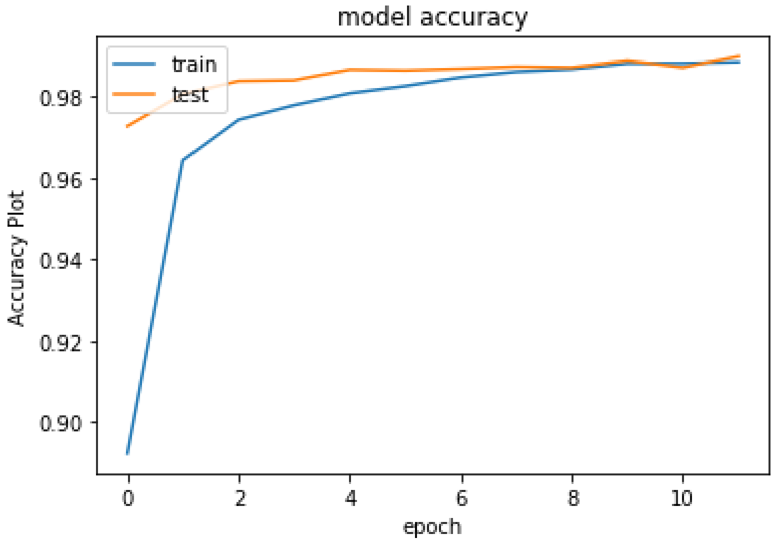
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(2, 2),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, 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'])

history = model.fit(x_train, y_train,
          batch_size=batch_size,
           epochs=epochs,
           verbose=1,
           validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

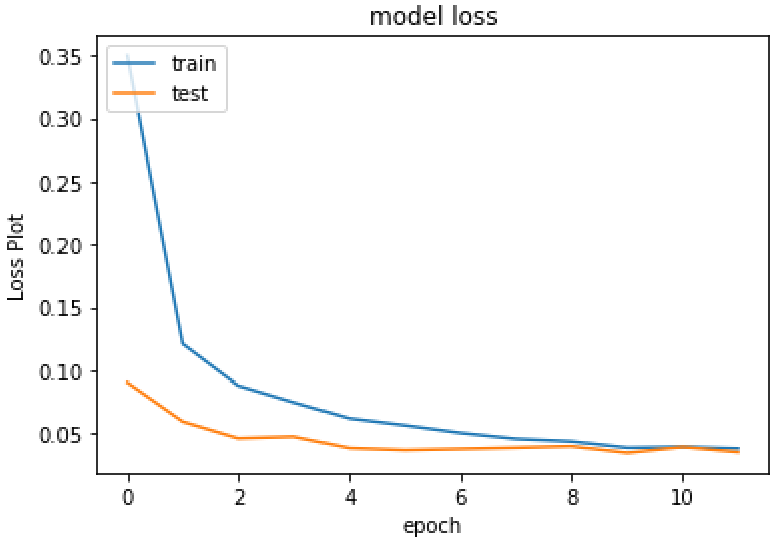
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 116s 2ms/step - loss: 0.
3501 - acc: 0.8922 - val_loss: 0.0902 - val_acc: 0.9728
Epoch 2/12
60000/60000 [==============================] - 114s 2ms/step - loss: 0.
1209 - acc: 0.9643 - val_loss: 0.0592 - val_acc: 0.9807
Epoch 3/12
60000/60000 [==============================] - 115s 2ms/step - loss: 0.
0877 - acc: 0.9744 - val_loss: 0.0459 - val_acc: 0.9839
Epoch 4/12
60000/60000 [==============================] - 114s 2ms/step - loss: 0.
0745 - acc: 0.9779 - val_loss: 0.0472 - val_acc: 0.9841
Epoch 5/12
60000/60000 [==============================] - 114s 2ms/step - loss: 0.
0617 - acc: 0.9809 - val_loss: 0.0381 - val_acc: 0.9867
Epoch 6/12
60000/60000 [==============================] - 114s 2ms/step - loss: 0.
0562 - acc: 0.9827 - val_loss: 0.0366 - val_acc: 0.9865
Epoch 7/12
60000/60000 [==============================] - 114s 2ms/step - loss: 0.
0502 - acc: 0.9848 - val_loss: 0.0375 - val_acc: 0.9869
Epoch 8/12
60000/60000 [==============================] - 114s 2ms/step - loss: 0.
0455 - acc: 0.9862 - val_loss: 0.0385 - val_acc: 0.9874
Epoch 9/12
60000/60000 [==============================] - 116s 2ms/step - loss: 0.
0435 - acc: 0.9868 - val_loss: 0.0396 - val_acc: 0.9872
Epoch 10/12
60000/60000 [==============================] - 116s 2ms/step - loss: 0.
0384 - acc: 0.9882 - val_loss: 0.0344 - val_acc: 0.9889
Epoch 11/12

60000/60000 [==============================] - 116s 2ms/step - loss: 0.
```

```
60000/60000 [==============================] - 116s 2ms/step - loss: 0.
0391 - acc: 0.9882 - val_loss: 0.0391 - val_acc: 0.9872
Epoch 12/12
60000/60000 [==============================] - 115s 2ms/step - loss: 0.
0378 - acc: 0.9885 - val_loss: 0.0350 - val_acc: 0.9900
Test loss: 0.03499671397599209
Test accuracy: 0.99
```

In [0]:

```
#  Accuracy of the model
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('Accuracy Plot')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# Loss of the model
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('Loss Plot')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```





CNN model using 5*5 Kernel

In [0]:
```
model = Sequential()
```

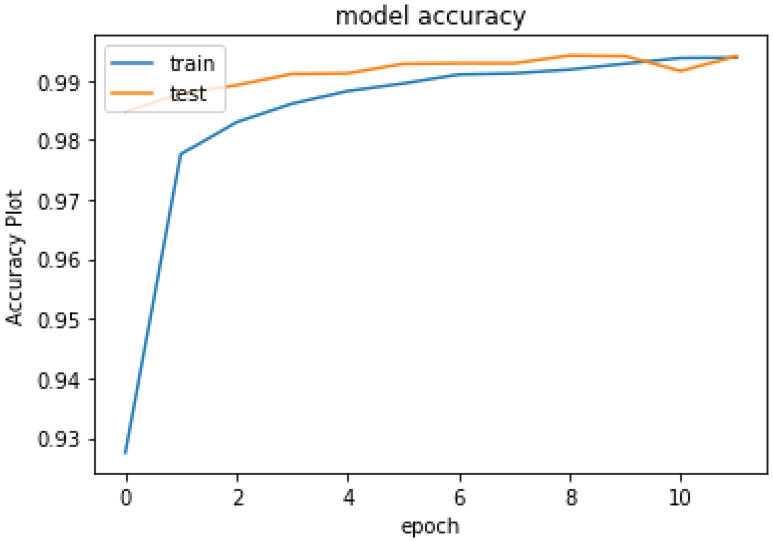
```
model.add(Conv2D(32, kernel_size=(5, 5),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
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'])

history = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

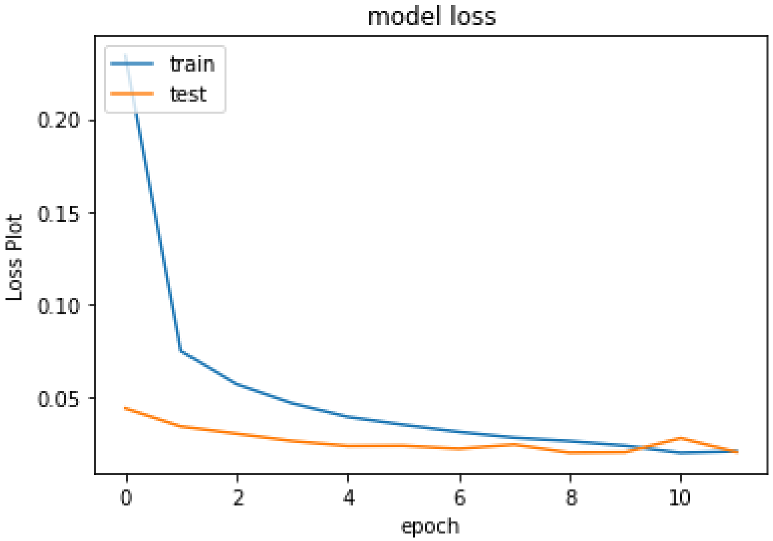
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 210s 4ms/step - loss: 0.
2339 - acc: 0.9275 - val_loss: 0.0439 - val_acc: 0.9847
Epoch 2/12
60000/60000 [==============================] - 210s 3ms/step - loss: 0.
0750 - acc: 0.9776 - val_loss: 0.0341 - val_acc: 0.9878
Epoch 3/12
60000/60000 [==============================] - 209s 3ms/step - loss: 0.
0570 - acc: 0.9830 - val_loss: 0.0302 - val_acc: 0.9892
Epoch 4/12
60000/60000 [==============================] - 209s 3ms/step - loss: 0.
0467 - acc: 0.9861 - val_loss: 0.0262 - val_acc: 0.9911
Epoch 5/12
60000/60000 [==============================] - 208s 3ms/step - loss: 0.
0393 - acc: 0.9882 - val_loss: 0.0235 - val_acc: 0.9912
```

```
Epoch 6/12
60000/60000 [==============================] - 209s 3ms/step - loss: 0.
0350 - acc: 0.9895 - val_loss: 0.0237 - val_acc: 0.9928
Epoch 7/12
60000/60000 [==============================] - 209s 3ms/step - loss: 0.
0311 - acc: 0.9910 - val_loss: 0.0220 - val_acc: 0.9929
Epoch 8/12
60000/60000 [==============================] - 209s 3ms/step - loss: 0.
0280 - acc: 0.9912 - val_loss: 0.0243 - val_acc: 0.9929
Epoch 9/12
60000/60000 [==============================] - 209s 3ms/step - loss: 0.
0262 - acc: 0.9918 - val_loss: 0.0199 - val_acc: 0.9942
Epoch 10/12
60000/60000 [==============================] - 209s 3ms/step - loss: 0.
0236 - acc: 0.9928 - val_loss: 0.0202 - val_acc: 0.9941
Epoch 11/12
60000/60000 [==============================] - 208s 3ms/step - loss: 0.
0198 - acc: 0.9938 - val_loss: 0.0278 - val_acc: 0.9916
Epoch 12/12
60000/60000 [==============================] - 209s 3ms/step - loss: 0.
0206 - acc: 0.9939 - val_loss: 0.0202 - val_acc: 0.9941
Test loss: 0.020204613963087103
Test accuracy: 0.9941
```

In [0]:

```
#  Accuracy of the model
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('Accuracy Plot')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# Loss of the mode
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('Loss Plot')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```





CNN model using 3 layers and 3*3 Kernel size

In [0]:

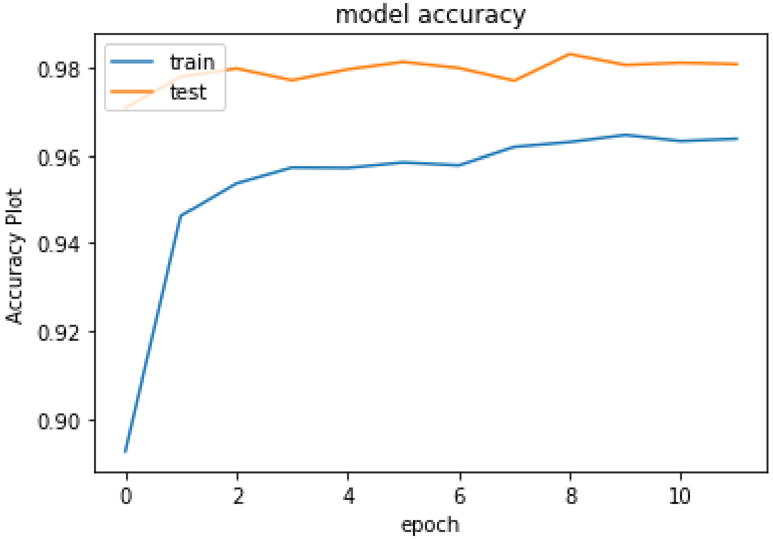
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.30))

model.add(Conv2D(128, (3, 3),
                 activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.40))

model.add(Flatten())
model.add(Dense(512, 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'])

history = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

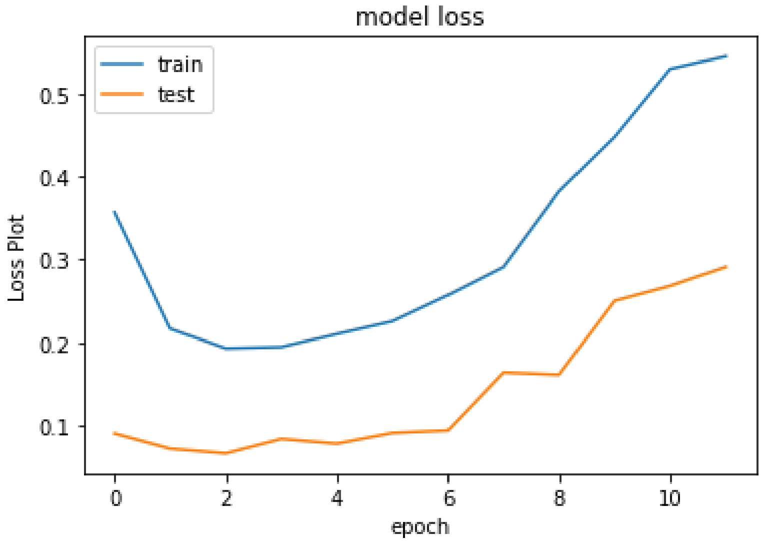
```
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 209s 3ms/step - loss: 0.
3570 - acc: 0.8926 - val_loss: 0.0904 - val_acc: 0.9709
Epoch 2/12
60000/60000 [==============================] - 209s 3ms/step - loss: 0.
```

```
2170 - acc: 0.9462 - val_loss: 0.0722 - val_acc: 0.9780
Epoch 3/12
60000/60000 [==============================] - 208s 3ms/step - loss: 0.
1928 - acc: 0.9537 - val_loss: 0.0666 - val_acc: 0.9799
Epoch 4/12
60000/60000 [==============================] - 208s 3ms/step - loss: 0.
1945 - acc: 0.9573 - val_loss: 0.0838 - val_acc: 0.9772
Epoch 5/12
60000/60000 [==============================] - 208s 3ms/step - loss: 0.
2105 - acc: 0.9572 - val_loss: 0.0782 - val_acc: 0.9797
Epoch 6/12
60000/60000 [==============================] - 207s 3ms/step - loss: 0.
2258 - acc: 0.9585 - val_loss: 0.0910 - val_acc: 0.9814
Epoch 7/12
60000/60000 [==============================] - 207s 3ms/step - loss: 0.
2571 - acc: 0.9578 - val_loss: 0.0940 - val_acc: 0.9800
Epoch 8/12
60000/60000 [==============================] - 207s 3ms/step - loss: 0.
2909 - acc: 0.9619 - val_loss: 0.1640 - val_acc: 0.9771
Epoch 9/12
60000/60000 [==============================] - 207s 3ms/step - loss: 0.
3819 - acc: 0.9630 - val_loss: 0.1612 - val_acc: 0.9832
Epoch 10/12
60000/60000 [==============================] - 207s 3ms/step - loss: 0.
4476 - acc: 0.9647 - val_loss: 0.2505 - val_acc: 0.9807
Epoch 11/12
60000/60000 [==============================] - 207s 3ms/step - loss: 0.
5293 - acc: 0.9633 - val_loss: 0.2683 - val_acc: 0.9812
Epoch 12/12
60000/60000 [==============================] - 206s 3ms/step - loss: 0.
5447 - acc: 0.9638 - val_loss: 0.2912 - val_acc: 0.9809
Test loss: 0.2911788964957141
Test accuracy: 0.9809
```

In [0]:

```
#  Accuracy of the model
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('Accuracy Plot')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# Loss of the model
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('Loss Plot')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```





CNN model using 5 layers and 3*3 Kernel size

In [0]:

```
model = Sequential()
print(input_shape)
model.add(Conv2D(32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=input_shape))

model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.30))

model.add(Conv2D(64, (3, 3),
                 activation='relu', padding = 'same'))
model.add(Conv2D(64, (3, 3), activation='relu', padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.50))
model.add(Conv2D(128, (3, 3),
                 activation='relu'))

model.add(Flatten())
# Dense is used to make Fully connected layer between previous and next
 layer
model.add(Dense(256, 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'])

history = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

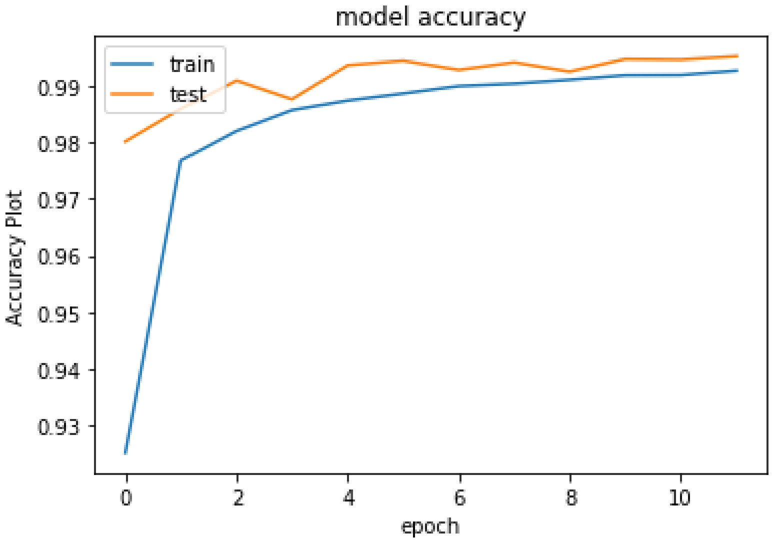
```
(28, 28, 1)
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:2041: The name tf.nn.fused_batch_norm is de
precated. Please use tf.compat.v1.nn.fused_batch_norm instead.

Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 185s 3ms/step - loss: 0.
2470 - acc: 0.9251 - val_loss: 0.0626 - val_acc: 0.9802
Epoch 2/12
60000/60000 [==============================] - 184s 3ms/step - loss: 0.
0818 - acc: 0.9768 - val_loss: 0.0487 - val_acc: 0.9860
Epoch 3/12
60000/60000 [==============================] - 184s 3ms/step - loss: 0.
0616 - acc: 0.9821 - val_loss: 0.0278 - val_acc: 0.9910
Epoch 4/12
```

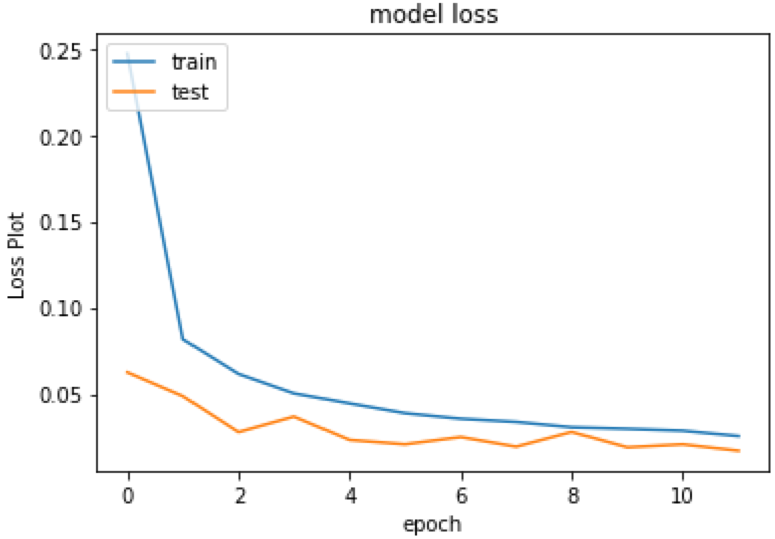
```
60000/60000 [==============================] - 184s 3ms/step - loss: 0.
0503 - acc: 0.9858 - val_loss: 0.0368 - val_acc: 0.9877
Epoch 5/12
60000/60000 [==============================] - 183s 3ms/step - loss: 0.
0445 - acc: 0.9875 - val_loss: 0.0232 - val_acc: 0.9937
Epoch 6/12
60000/60000 [==============================] - 184s 3ms/step - loss: 0.
0388 - acc: 0.9887 - val_loss: 0.0207 - val_acc: 0.9945
Epoch 7/12
60000/60000 [==============================] - 183s 3ms/step - loss: 0.
0356 - acc: 0.9900 - val_loss: 0.0249 - val_acc: 0.9929
Epoch 8/12
60000/60000 [==============================] - 184s 3ms/step - loss: 0.
0337 - acc: 0.9905 - val_loss: 0.0193 - val_acc: 0.9942
Epoch 9/12
60000/60000 [==============================] - 184s 3ms/step - loss: 0.
0306 - acc: 0.9911 - val_loss: 0.0279 - val_acc: 0.9926
Epoch 10/12
60000/60000 [==============================] - 184s 3ms/step - loss: 0.
0297 - acc: 0.9920 - val_loss: 0.0190 - val_acc: 0.9948
Epoch 11/12
60000/60000 [==============================] - 184s 3ms/step - loss: 0.
0286 - acc: 0.9920 - val_loss: 0.0207 - val_acc: 0.9947
Epoch 12/12
60000/60000 [==============================] - 184s 3ms/step - loss: 0.
0255 - acc: 0.9928 - val_loss: 0.0170 - val_acc: 0.9953
Test loss: 0.016960171616025 72
Test accuracy: 0.9953
```

In [0]:

```python
import matplotlib.pyplot as plt
#  "Accuracy"
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('Accuracy Plot')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

```
# "Loss"
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('Loss Plot')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```





CNN model using 7 layers and 2*2 Kernel size

In [6]:

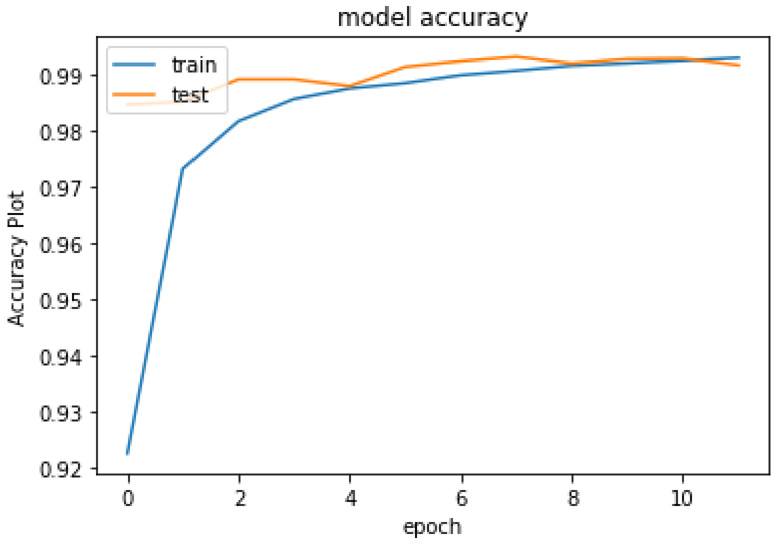
```
model = Sequential()
model.add(Conv2D(32, kernel_size=(2, 2),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(32, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.30))

model.add(Conv2D(64, (2, 2), activation='relu'))
model.add(Conv2D(64, (2, 2), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(BatchNormalization())
model.add(Dropout(0.50))

model.add(Conv2D(128, (2, 2),
                 activation='relu'))
model.add(Conv2D(128, (2, 2), activation='relu'))
model.add(BatchNormalization())
# model.add(MaxPooling2D(pool_size=(2, 2)))


model.add(Conv2D(256, (3, 3),
                 activation='relu', padding = 'same'))
# model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Dropout(0.30))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
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'])

history = model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

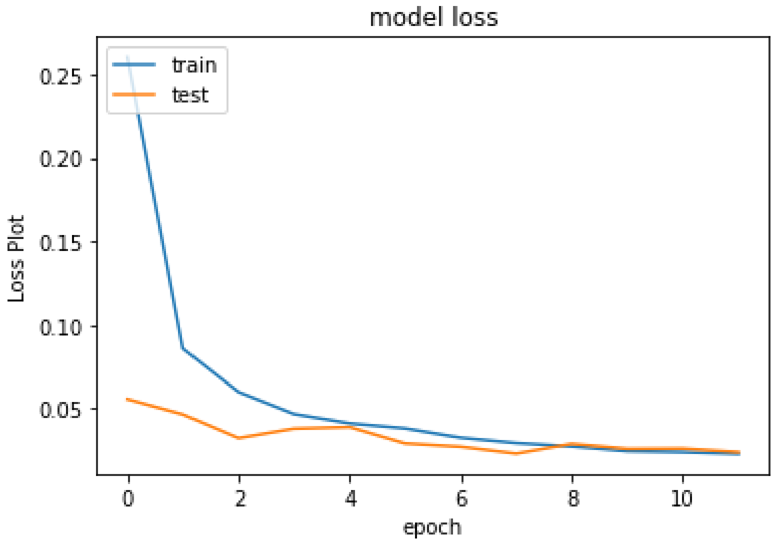
```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/op
timizers.py:793: The name tf.train.Optimizer is deprecated. Please use
tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:3576: The name tf.log is deprecated. Please
use tf.math.log instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorfl
ow_core/python/ops/math_grad.py:1424: where (from tensorflow.python.op
s.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
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:1033: The name tf.assign_add is deprecated.
Please use tf.compat.v1.assign_add instead.

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/ba
ckend/tensorflow_backend.py:1020: The name tf.assign is deprecated. Ple
ase use tf.compat.v1.assign instead.

Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 192s 3ms/step - loss: 0.
2606 - acc: 0.9227 - val_loss: 0.0556 - val_acc: 0.9848
Epoch 2/12
60000/60000 [==============================] - 191s 3ms/step - loss: 0.
0860 - acc: 0.9733 - val_loss: 0.0465 - val_acc: 0.9853
Epoch 3/12
60000/60000 [==============================] - 191s 3ms/step - loss: 0.
0599 - acc: 0.9818 - val_loss: 0.0323 - val_acc: 0.9893
Epoch 4/12
60000/60000 [==============================] - 191s 3ms/step - loss: 0.
0467 - acc: 0.9858 - val_loss: 0.0380 - val_acc: 0.9893
Epoch 5/12
60000/60000 [==============================] - 191s 3ms/step - loss: 0.
0411 - acc: 0.9877 - val_loss: 0.0389 - val_acc: 0.9881
Epoch 6/12
60000/60000 [==============================] - 191s 3ms/step - loss: 0.
0381 - acc: 0.9886 - val_loss: 0.0291 - val_acc: 0.9915
Epoch 7/12
60000/60000 [==============================] - 191s 3ms/step - loss: 0.
0326 - acc: 0.9900 - val_loss: 0.0272 - val_acc: 0.9925
Epoch 8/12
60000/60000 [==============================] - 191s 3ms/step - loss: 0.
0294 - acc: 0.9908 - val_loss: 0.0231 - val_acc: 0.9933
Epoch 9/12
60000/60000 [==============================] - 191s 3ms/step - loss: 0.
0274 - acc: 0.9917 - val_loss: 0.0289 - val_acc: 0.9921
```

```
Epoch 10/12
60000/60000 [==============================] - 190s 3ms/step - loss: 0.
0245 - acc: 0.9921 - val_loss: 0.0259 - val_acc: 0.9929
Epoch 11/12
60000/60000 [==============================] - 190s 3ms/step - loss: 0.
0239 - acc: 0.9926 - val_loss: 0.0261 - val_acc: 0.9930
Epoch 12/12
60000/60000 [==============================] - 190s 3ms/step - loss: 0.
0229 - acc: 0.9931 - val_loss: 0.0239 - val_acc: 0.9918
Test loss: 0.02392568226782496
Test accuracy: 0.9918
```

In [11]:

```
import matplotlib.pyplot as plt
#  "Accuracy"
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('Accuracy Plot')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# "Loss"
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('Loss Plot')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```





In [15]:
```
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Model_Architecture", "Test loss", "Test Accuracy"]
```

```
x.add_row(["CNN model using 2*2 Kernel", 0.0349, 0.99])
x.add_row(["CNN model using 3*3 Kernel", 0.0278, 0.9921])
x.add_row(["CNN model using 5*5 Kernel", 0.0202, 0.9941])
x.add_row(["CNN model using 3 layers and 3*3 Kernel size", 0.2911, 0.98
09])
x.add_row(["CNN model using 5 layers and 3*3 Kernel size", 0.0169, 0.99
53])
x.add_row(["CNN model using 7 layers and 2*2 Kernel size", 0.0239, 0.99
18])
print(x)
```

```
+----------------------------------------------+-----------+-----------
----+
|              Model_Architecture              | Test loss | Test Accur
acy |
+----------------------------------------------+-----------+-----------
----+
|          CNN model using 2*2 Kernel          |   0.0349  |      0.99
    |
|          CNN model using 3*3 Kernel          |   0.0278  |     0.9921
    |
|          CNN model using 5*5 Kernel          |   0.0202  |     0.9941
    |
| CNN model using 3 layers and 3*3 Kernel size |   0.2911  |     0.9809
    |
| CNN model using 5 layers and 3*3 Kernel size |   0.0169  |     0.9953
    |
| CNN model using 7 layers and 2*2 Kernel size |   0.0239  |     0.9918
    |
+----------------------------------------------+-----------+-----------
----+
```

In [0]: