In [2]:

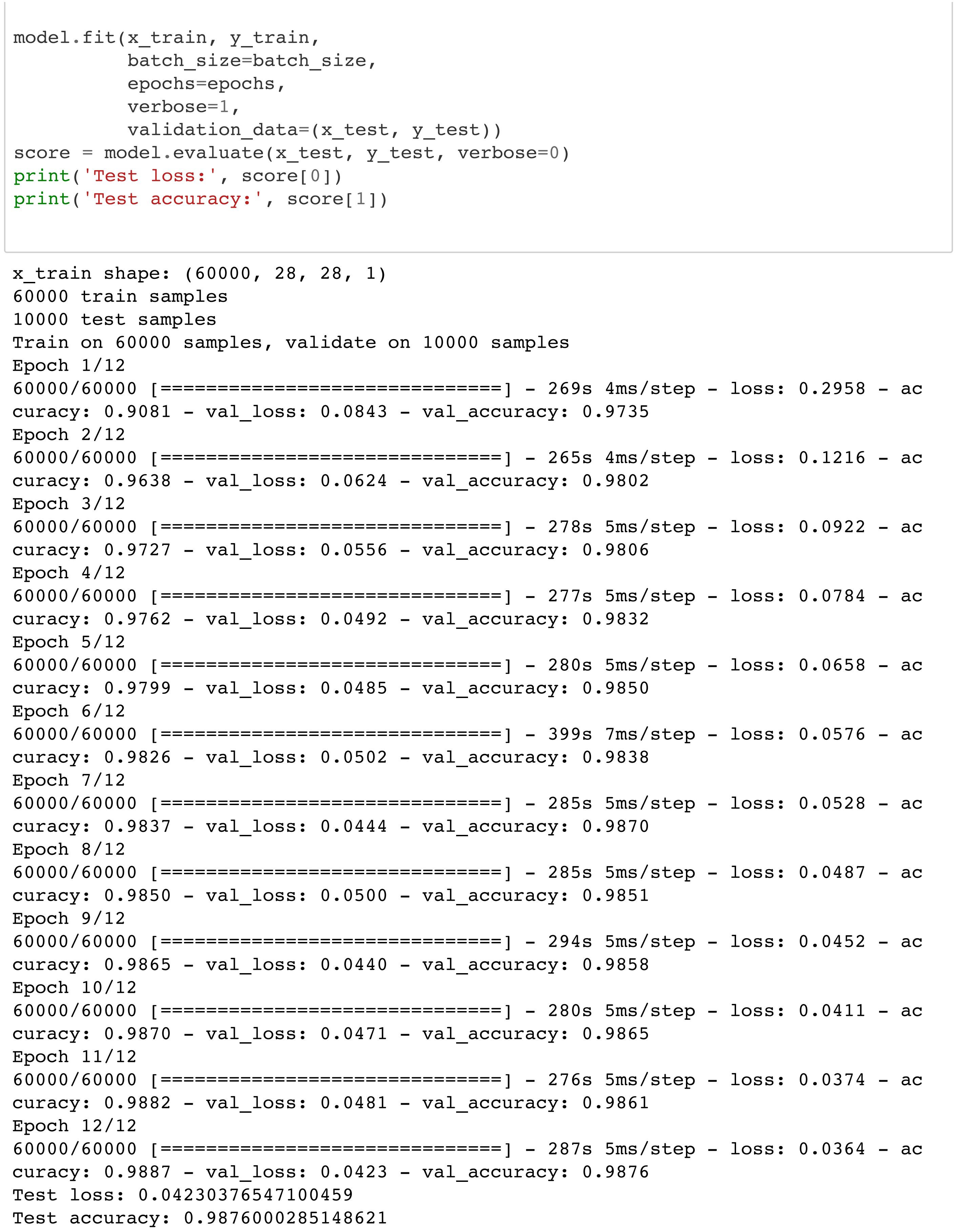
```
# Credits: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py


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

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)

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'])

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])
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 236s 4ms/step - loss: 0.2797 - ac
curacy: 0.9136 - val_loss: 0.0651 - val_accuracy: 0.9783
Epoch 2/12
60000/60000 [==============================] - 238s 4ms/step - loss: 0.0911 - ac
curacy: 0.9729 - val_loss: 0.0411 - val_accuracy: 0.9867
Epoch 3/12
60000/60000 [==============================] - 231s 4ms/step - loss: 0.0656 - ac
curacy: 0.9807 - val_loss: 0.0411 - val_accuracy: 0.9872
Epoch 4/12
60000/60000 [==============================] - 218s 4ms/step - loss: 0.0553 - ac
curacy: 0.9836 - val_loss: 0.0320 - val_accuracy: 0.9894
Epoch 5/12
60000/60000 [==============================] - 216s 4ms/step - loss: 0.0482 - ac
curacy: 0.9857 - val_loss: 0.0383 - val_accuracy: 0.9873
Epoch 6/12
60000/60000 [==============================] - 217s 4ms/step - loss: 0.0421 - ac
curacy: 0.9873 - val_loss: 0.0287 - val_accuracy: 0.9907
Epoch 7/12
60000/60000 [==============================] - 216s 4ms/step - loss: 0.0380 - ac
curacy: 0.9882 - val_loss: 0.0322 - val_accuracy: 0.9901
Epoch 8/12
60000/60000 [==============================] - 215s 4ms/step - loss: 0.0362 - ac
curacy: 0.9891 - val_loss: 0.0305 - val_accuracy: 0.9907
Epoch 9/12
60000/60000 [==============================] - 217s 4ms/step - loss: 0.0324 - ac
curacy: 0.9901 - val_loss: 0.0280 - val_accuracy: 0.9905
Epoch 10/12
60000/60000 [==============================] - 223s 4ms/step - loss: 0.0295 - ac
curacy: 0.9912 - val_loss: 0.0273 - val_accuracy: 0.9910
Epoch 11/12
60000/60000 [==============================] - 215s 4ms/step - loss: 0.0310 - ac
curacy: 0.9905 - val_loss: 0.0267 - val_accuracy: 0.9913
Epoch 12/12
60000/60000 [==============================] - 221s 4ms/step - loss: 0.0279 - ac
curacy: 0.9915 - val_loss: 0.0305 - val_accuracy: 0.9900
Test loss: 0.03053935998292145
Test accuracy: 0.9900000095367432
```

Adding another Conv Layer - total 3 layers

In [3]:

```
# Credits: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py


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

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)

model = Sequential()
model.add(Conv2D(32, kernel_size=(1, 1),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (1, 1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))

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'])
```



```
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])
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 269s 4ms/step - loss: 0.2958 - ac
curacy: 0.9081 - val_loss: 0.0843 - val_accuracy: 0.9735
Epoch 2/12
60000/60000 [==============================] - 265s 4ms/step - loss: 0.1216 - ac
curacy: 0.9638 - val_loss: 0.0624 - val_accuracy: 0.9802
Epoch 3/12
60000/60000 [==============================] - 278s 5ms/step - loss: 0.0922 - ac
curacy: 0.9727 - val_loss: 0.0556 - val_accuracy: 0.9806
Epoch 4/12
60000/60000 [==============================] - 277s 5ms/step - loss: 0.0784 - ac
curacy: 0.9762 - val_loss: 0.0492 - val_accuracy: 0.9832
Epoch 5/12
60000/60000 [==============================] - 280s 5ms/step - loss: 0.0658 - ac
curacy: 0.9799 - val_loss: 0.0485 - val_accuracy: 0.9850
Epoch 6/12
60000/60000 [==============================] - 399s 7ms/step - loss: 0.0576 - ac
curacy: 0.9826 - val_loss: 0.0502 - val_accuracy: 0.9838
Epoch 7/12
60000/60000 [==============================] - 285s 5ms/step - loss: 0.0528 - ac
curacy: 0.9837 - val_loss: 0.0444 - val_accuracy: 0.9870
Epoch 8/12
60000/60000 [==============================] - 285s 5ms/step - loss: 0.0487 - ac
curacy: 0.9850 - val_loss: 0.0500 - val_accuracy: 0.9851
Epoch 9/12
60000/60000 [==============================] - 294s 5ms/step - loss: 0.0452 - ac
curacy: 0.9865 - val_loss: 0.0440 - val_accuracy: 0.9858
Epoch 10/12
60000/60000 [==============================] - 280s 5ms/step - loss: 0.0411 - ac
curacy: 0.9870 - val_loss: 0.0471 - val_accuracy: 0.9865
Epoch 11/12
60000/60000 [==============================] - 276s 5ms/step - loss: 0.0374 - ac
curacy: 0.9882 - val_loss: 0.0481 - val_accuracy: 0.9861
Epoch 12/12
60000/60000 [==============================] - 287s 5ms/step - loss: 0.0364 - ac
curacy: 0.9887 - val_loss: 0.0423 - val_accuracy: 0.9876
Test loss: 0.04230376547100459
Test accuracy: 0.9876000285148621
```

Adding another Conv Layer - total 5 layers

In [4]:

```python
# Credits: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py


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

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)

model = Sequential()
model.add(Conv2D(32, kernel_size=(1, 1),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (1, 1), activation='relu'))
model.add(Conv2D(64, (1, 1), activation='relu'))
model.add(Conv2D(64, (1, 1), activation='relu'))
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'])

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])
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 579s 10ms/step - loss: 0.2890 - a
ccuracy: 0.9118 - val_loss: 0.0728 - val_accuracy: 0.9763
Epoch 2/12
60000/60000 [==============================] - 581s 10ms/step - loss: 0.1025 - a
ccuracy: 0.9705 - val_loss: 0.0488 - val_accuracy: 0.9845
Epoch 3/12
60000/60000 [==============================] - 576s 10ms/step - loss: 0.0775 - a
ccuracy: 0.9768 - val_loss: 0.0489 - val_accuracy: 0.9846
Epoch 4/12
60000/60000 [==============================] - 574s 10ms/step - loss: 0.0643 - a
ccuracy: 0.9806 - val_loss: 0.0421 - val_accuracy: 0.9872
Epoch 5/12
60000/60000 [==============================] - 581s 10ms/step - loss: 0.0538 - a
ccuracy: 0.9843 - val_loss: 0.0414 - val_accuracy: 0.9869
Epoch 6/12
60000/60000 [==============================] - 575s 10ms/step - loss: 0.0471 - a
ccuracy: 0.9861 - val_loss: 0.0442 - val_accuracy: 0.9855
Epoch 7/12
60000/60000 [==============================] - 580s 10ms/step - loss: 0.0417 - a
ccuracy: 0.9873 - val_loss: 0.0476 - val_accuracy: 0.9878
Epoch 8/12
60000/60000 [==============================] - 615s 10ms/step - loss: 0.0385 - a
ccuracy: 0.9884 - val_loss: 0.0406 - val_accuracy: 0.9889
Epoch 9/12
60000/60000 [==============================] - 595s 10ms/step - loss: 0.0338 - a
ccuracy: 0.9893 - val_loss: 0.0370 - val_accuracy: 0.9892
Epoch 10/12
60000/60000 [==============================] - 751s 13ms/step - loss: 0.0316 - a
ccuracy: 0.9902 - val_loss: 0.0387 - val_accuracy: 0.9889
Epoch 11/12
60000/60000 [==============================] - 644s 11ms/step - loss: 0.0286 - a
ccuracy: 0.9908 - val_loss: 0.0418 - val_accuracy: 0.9887
Epoch 12/12
60000/60000 [==============================] - 610s 10ms/step - loss: 0.0273 - a
ccuracy: 0.9914 - val_loss: 0.0397 - val_accuracy: 0.9895
Test loss: 0.039656925907363985
Test accuracy: 0.9894999861717224
```

Adding another Conv Layer - total 7 layers

In [5]:

```
# Credits: https://github.com/keras-team/keras/blob/master/examples/mnist_cnn.py


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

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)

model = Sequential()
model.add(Conv2D(32, kernel_size=(1, 1),
                 activation='relu',
                 input_shape=input_shape))
model.add(Conv2D(64, (1, 1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (1, 1), activation='relu'))
model.add(Conv2D(64, (1, 1), activation='relu'))
model.add(Conv2D(64, (1, 1), activation='relu'))
model.add(Conv2D(64, (1, 1), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(Dropout(0.5))
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'])

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])
```

```
x_train shape: (60000, 28, 28, 1)
60000 train samples
10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/12
60000/60000 [==============================] - 291s 5ms/step - loss: 0.3091 - ac
curacy: 0.9050 - val_loss: 0.0839 - val_accuracy: 0.9731
Epoch 2/12
60000/60000 [==============================] - 283s 5ms/step - loss: 0.1308 - ac
curacy: 0.9607 - val_loss: 0.0656 - val_accuracy: 0.9797
Epoch 3/12
60000/60000 [==============================] - 285s 5ms/step - loss: 0.1059 - ac
curacy: 0.9685 - val_loss: 0.0722 - val_accuracy: 0.9777
Epoch 4/12
60000/60000 [==============================] - 299s 5ms/step - loss: 0.0924 - ac
curacy: 0.9721 - val_loss: 0.0470 - val_accuracy: 0.9846
Epoch 5/12
60000/60000 [==============================] - 284s 5ms/step - loss: 0.0842 - ac
curacy: 0.9747 - val_loss: 0.0473 - val_accuracy: 0.9838
Epoch 6/12
60000/60000 [==============================] - 299s 5ms/step - loss: 0.0758 - ac
curacy: 0.9772 - val_loss: 0.0466 - val_accuracy: 0.9856
Epoch 7/12
60000/60000 [==============================] - 402s 7ms/step - loss: 0.0707 - ac
curacy: 0.9790 - val_loss: 0.0418 - val_accuracy: 0.9866
Epoch 8/12
60000/60000 [==============================] - 299s 5ms/step - loss: 0.0664 - ac
curacy: 0.9797 - val_loss: 0.0465 - val_accuracy: 0.9854
Epoch 9/12
60000/60000 [==============================] - 308s 5ms/step - loss: 0.0624 - ac
curacy: 0.9815 - val_loss: 0.0372 - val_accuracy: 0.9878
Epoch 10/12
60000/60000 [==============================] - 299s 5ms/step - loss: 0.0586 - ac
curacy: 0.9820 - val_loss: 0.0416 - val_accuracy: 0.9875
Epoch 11/12
60000/60000 [==============================] - 346s 6ms/step - loss: 0.0566 - ac
curacy: 0.9834 - val_loss: 0.0379 - val_accuracy: 0.9884
Epoch 12/12
60000/60000 [==============================] - 387s 6ms/step - loss: 0.0564 - ac
curacy: 0.9824 - val_loss: 0.0418 - val_accuracy: 0.9878
Test loss: 0.04180917171128975
Test accuracy: 0.9878000020980835
```

In [ ]:

```

```