```
from __future__ import print_function
from numpy import random
random.seed(42)  # @UndefinedVariable

from keras.datasets import mnist
from keras.models import Model
from keras.layers import Input, Add, Dense, Activation, ZeroPadding2D,
BatchNormalization, Flatten, Conv2D, AveragePooling2D, MaxPooling2D
from keras.layers import Convolution2D, MaxPooling2D
from keras.utils import np_utils
from keras import backend as K

batch_size = 200
nb_classes = 10
nb_epoch = 5

img_rows, img_cols = 28, 28
pool_size = (2, 2)
kernel_size = (3, 3)

(X_train, y_train), (X_test, y_test) = mnist.load_data()

if keras.backend.image_data_format() == 'th':
    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 = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)

input_var = Input(shape=input_shape)

conv1 = Convolution2D(64, kernel_size[0], kernel_size[1],
                      border_mode='same', activation='relu')
(input_var)

print(conv1)

conv2 = Convolution2D(32, kernel_size[0], kernel_size[1],
                      border_mode='same', activation='relu')(conv1)

print (conv2)
```

```
resnet1 = Add()([input_var, conv2])
  resnet1 = Activation('relu')(resnet1)

print (resnet1)

conv3 = Convolution2D(8, kernel_size[0], kernel_size[1],
                      border_mode='same', activation='relu')(resnet1)

print (conv3)

conv4 = Convolution2D(16, kernel_size[0], kernel_size[1],
                      border_mode='same', activation='relu')(conv3)

print (conv4)

resnet2 = Add()([resnet1, conv4])
resnet2 = Activation('relu')(resnet2)

print (resnet2)

conv5 = Convolution2D(8, kernel_size[0], kernel_size[1],
                      border_mode='same', activation='relu')(resnet2)

print (conv5)

conv6 = Convolution2D(16, kernel_size[0], kernel_size[1],
                      border_mode='same', activation='relu')(conv5)

print (conv6)

resnet3=Add()([resnet2,conv6])
resnet3=Activation('relu')(resnet3)
print (resnet3)

mxpool = MaxPooling2D(pool_size=pool_size)(resnet)
flat = Flatten()(mxpool)
dropout = Dropout(0.5)(flat)
softmax = Dense(nb_classes, activation='softmax')(dropout)

print (softmax)

model = Model(input=[input_var], output=[softmax])
model.compile(loss='categorical_crossentropy',
              optimizer='adadelta',
              metrics=['accuracy'])

model.fit(X_train, Y_train, batch_size=batch_size, nb_epoch=nb_epoch,
          verbose=1, validation_data=(X_test, Y_test))
model.save('mnist_model.h5')
score = model.evaluate(X_test, Y_test, verbose=0)
print('Test score:', score[0])
print('Test accuracy:', score[1])
```