A、CNN 卷积神经网络

```
from keras.datasets import mnist
from keras.utils import np_utils
from keras.layers import Dense, Activation,
Convolution2D, MaxPooling2D, Flatten
from keras.optimizers import Adam
from keras import Sequential
import numpy as np

(X_train, y_train), (X_test, y_test) =
mnist.load_data()
X_train = X_train.reshape(-1, 1, 28, 28)
X_test = X_test.reshape(-1, 1, 28, 28)
y_train = np_utils.to_categorical(y_train, 10)
y_test = np_utils.to_categorical(y_test, 10)

model = Sequential(
    [
        Convolution2D(
            filters=32,
            kernel_size=(5, 5),
            padding='same',
            input_shape=(1, 28, 28)
        ),
        Activation('relu'),
        MaxPooling2D(
            pool_size=(2, 2),
            strides=(2, 2),
            padding='same'
        ),
        Convolution2D(
            filters=64,
            kernel_size=(5, 5),
            padding='same'
        ),
        Activation('relu'),
        MaxPooling2D(
            pool_size=(2, 2),
            padding='same'
        ),
        Flatten(),
        Dense(1024),
        Activation('relu'),
```

```
        Dense(10),
        Activation('softmax')
    ]
)

adam = Adam(lr=1e-4)
model.compile(optimizer=adam,
loss='categorical_crossentropy', metrics=['accuracy'])

print('Training')
model.fit(X_train, y_train, epochs=1, batch_size=32)

print('Testing')
loss, accuracy = model.evaluate(X_test, y_test)
print('loss:', loss)
print('accuracy', accuracy)
```

B、RNN 循环神经网络

```
from keras.datasets import mnist
from keras.utils import np_utils
from keras import Sequential
from keras.layers import SimpleRNN, Activation, Dense
from keras.optimizers import Adam

TIME_STEPS = 28
INPUT_SIZE = 28
BATCH_SIZE = 50
BATCH_INDEX = 0
OUTPUT_SIZE = 10
CELL_SIZE = 50
LR = 0.001

(X_train, y_train), (X_test, y_test) =
mnist.load_data()
X_train = X_train.reshape(-1, 28, 28) / 255
X_test = X_test.reshape(-1, 28, 28) / 255
y_train = np_utils.to_categorical(y_train, 10)
y_test = np_utils.to_categorical(y_test, 10)

model = Sequential(
    [
        SimpleRNN(
```

```
            batch_input_shape=(BATCH_SIZE, TIME_STEPS,
INPUT_SIZE),
            units=CELL_SIZE
        ),
        Dense(OUTPUT_SIZE),
        Activation('softmax')
    ]
)

adam = Adam(LR)
model.compile(optimizer=adam,
loss='categorical_crossentropy', metrics=['accuracy'])

print('Training')
for step in range(4001):
    X_batch = X_train[BATCH_INDEX: BATCH_SIZE +
BATCH_INDEX, :, :]
    Y_batch = y_train[BATCH_INDEX: BATCH_SIZE +
BATCH_INDEX, :]
    cost = model.train_on_batch(X_batch, Y_batch)
    BATCH_INDEX += BATCH_SIZE
    BATCH_INDEX = 0 if BATCH_INDEX >= X_train.shape[0]
else BATCH_INDEX

    if step % 500 == 0:
        cost, accuracy = model.evaluate(X_test, y_test,
batch_size=y_test.shape[0], verbose=False)
        print('cost', cost,'accuracy', accuracy)
```