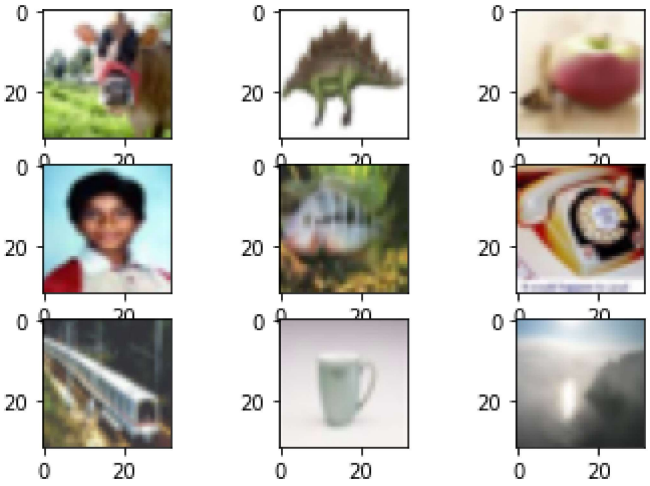
```
#Trần Nam Phương 20146470
from keras.datasets import cifar100
import matplotlib.pyplot as plt

(X_train,y_train),(X_test,y_test)=cifar100.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz
169009152/169001437 [==============================] - 2s 0us/step
169017344/169001437 [==============================] - 2s 0us/step
```

```
for i in range(9):
        plt.subplot(330+i+1)
        plt.imshow(X_train[i],cmap=plt.get_cmap('gray'))
plt.show()
```



```
X_train
```

```
...,
[170, 176, 150],
[161, 168, 130],
[146, 154, 113]],

[[255, 255, 255],
 [254, 254, 254],
 [255, 255, 255],

 ...,
 [189, 199, 169],
 [166, 178, 130],
 [121, 133,  87]],

...,

[[148, 185,  79],
 [142, 182,  57],
 [140, 179,  60],
 ...,
```

```
                [ 30,  17,   1],
                [ 65,  62,  15],
                [ 76,  77,  20]],

               [[122, 157,  66],
                [120, 155,  58],
                [126, 160,  71],
                ...,
                [ 22,  16,   3],
                [ 97, 112,  56],
                [141, 161,  87]],

               [[ 87, 122,  41],
                [ 88, 122,  39],
                [101, 134,  56],
                ...,
                [ 34,  36,  10],
                [105, 133,  59],
                [138, 173,  79]]],


              [[[255, 255, 255],
                [253, 253, 253],
                [253, 253, 253],
                ...,
                [253, 253, 253],
                [253, 253, 253],
                [255, 255, 255]],

               [[255, 255, 255],
                [255, 255, 255],
                [255, 255, 255],
                ...,
                [255, 255, 255],
                [255, 255, 255],
                [255, 255, 255]],

               [[255, 255, 255],
                [255, 255, 255],
```

```
X_train.shape
```

```
(50000, 32, 32, 3)
```

```
X_test.shape
```

```
(10000, 32, 32, 3)
```

```
X_train =X_train.reshape(50000, 3072)
X_test = X_test.reshape(10000, 3072)
```

```
X_train= X_train.astype('float32')
X_test=X_test.astype('float32')
```

```
X_train/=255
```

```
X_test /= 255
```

```
pip install np_utils
```

```
Collecting np_utils
  Downloading np_utils-0.6.0.tar.gz (61 kB)
     |████████████████████████████████| 61 kB 528 kB/s
Requirement already satisfied: numpy>=1.0 in /usr/local/lib/python3.7/dist-packages (fro
Building wheels for collected packages: np-utils
  Building wheel for np-utils (setup.py) ... done
  Created wheel for np-utils: filename=np_utils-0.6.0-py3-none-any.whl size=56459 sha256
  Stored in directory: /root/.cache/pip/wheels/d2/83/71/a781667865955ae7dc18e5a4038401de
Successfully built np-utils
Installing collected packages: np-utils
Successfully installed np-utils-0.6.0
```

```
y_train
```

```
array([[19],
       [29],
       [ 0],
       ...,
       [ 3],
       [ 7],
       [73]])
```

```
y_test
```

```
array([[49],
       [33],
       [72],
       ...,
       [51],
       [42],
       [70]])
```

```
from tensorflow.keras.utils import to_categorical
y_train=to_categorical(y_train, 100)
y_test=to_categorical(y_test, 100)
```

```
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense,Activation,Dropout

model=Sequential()
```

```
model.add(Dense(3872, activation='relu', input_shape=(3072,)))
model.add(Dropout(0.1))
model.add(Dense(1936, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(968, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(100, activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 3872)              11898656

 dropout (Dropout)           (None, 3872)              0

 dense_1 (Dense)             (None, 1936)              7498128

 dropout_1 (Dropout)         (None, 1936)              0

 dense_2 (Dense)             (None, 968)               1875016

 dropout_2 (Dropout)         (None, 968)               0

 dense_3 (Dense)             (None, 512)               496128

 dropout_3 (Dropout)         (None, 512)               0

 dense_4 (Dense)             (None, 512)               262656

 dropout_4 (Dropout)         (None, 512)               0

 dense_5 (Dense)             (None, 128)               65664

 dropout_5 (Dropout)         (None, 128)               0

 dense_6 (Dense)             (None, 100)               12900

=================================================================
Total params: 22,109,148
Trainable params: 22,109,148
Non-trainable params: 0
_________________________________________________________________
```

```
from tensorflow.keras.optimizers import RMSprop
model.compile(loss='categorical_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])
```

```
from keras.callbacks import EarlyStopping
history = model.fit(X_train,y_train,batch_size = 1000,epochs=20,verbose=1,validation_split=0.
```

```
Epoch 1/20
40/40 [==============================] - 97s 2s/step - loss: 5.6523 - accuracy: 0.0110 -
Epoch 2/20
40/40 [==============================] - 89s 2s/step - loss: 4.6073 - accuracy: 0.0117 -
Epoch 3/20
40/40 [==============================] - 89s 2s/step - loss: 4.6372 - accuracy: 0.0126 -
Epoch 4/20
40/40 [==============================] - 88s 2s/step - loss: 4.6229 - accuracy: 0.0146 -
Epoch 5/20
40/40 [==============================] - 88s 2s/step - loss: 4.5909 - accuracy: 0.0175 -
Epoch 6/20
40/40 [==============================] - 88s 2s/step - loss: 4.5169 - accuracy: 0.0224 -
Epoch 7/20
40/40 [==============================] - 88s 2s/step - loss: 4.4203 - accuracy: 0.0290 -
Epoch 8/20
40/40 [==============================] - 88s 2s/step - loss: 4.3428 - accuracy: 0.0359 -
Epoch 9/20
40/40 [==============================] - 88s 2s/step - loss: 4.2978 - accuracy: 0.0398 -
Epoch 10/20
40/40 [==============================] - 89s 2s/step - loss: 4.2335 - accuracy: 0.0480 -
Epoch 11/20
40/40 [==============================] - 89s 2s/step - loss: 4.1780 - accuracy: 0.0545 -
Epoch 12/20
40/40 [==============================] - 89s 2s/step - loss: 4.1524 - accuracy: 0.0578 -
Epoch 13/20
40/40 [==============================] - 89s 2s/step - loss: 4.1163 - accuracy: 0.0627 -
Epoch 14/20
40/40 [==============================] - 88s 2s/step - loss: 4.0810 - accuracy: 0.0678 -
Epoch 15/20
40/40 [==============================] - 89s 2s/step - loss: 4.0760 - accuracy: 0.0690 -
Epoch 16/20
40/40 [==============================] - 89s 2s/step - loss: 4.0254 - accuracy: 0.0768 -
Epoch 17/20
40/40 [==============================] - 89s 2s/step - loss: 4.0116 - accuracy: 0.0812 -
Epoch 18/20
40/40 [==============================] - 88s 2s/step - loss: 3.9773 - accuracy: 0.0863 -
Epoch 19/20
40/40 [==============================] - 87s 2s/step - loss: 3.9427 - accuracy: 0.0933 -
Epoch 20/20
40/40 [==============================] - 88s 2s/step - loss: 3.9416 - accuracy: 0.0947 -
```

```
model.save('B3.h5')
```

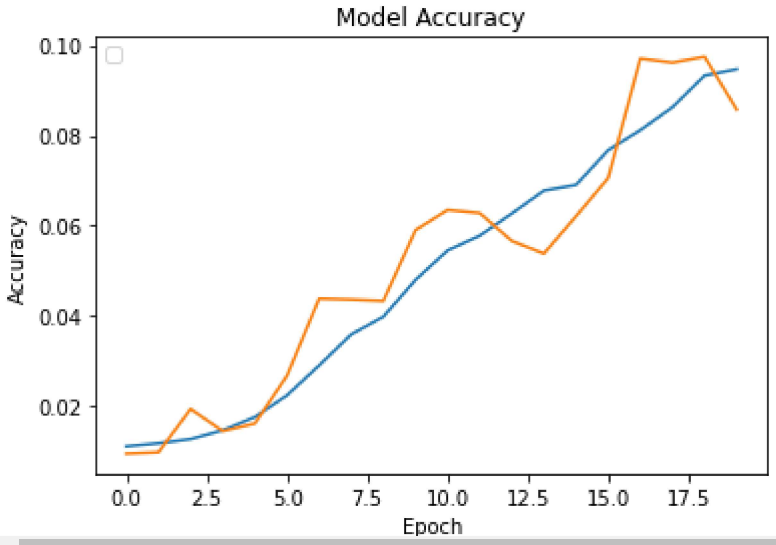
```
score=model.evaluate(X_test,y_test,verbose=0)
print('Test loss: ',score[0])
print('Test Accuracy: ',score[1])
```

```
Test loss:  4.02184534072876
```

```
Test Accuracy:  0.0885000005364418
```

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train'],['Validation'],loc='upper left')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: UserWarning: Legend does
A proxy artist may be used instead.
See: http://matplotlib.org/users/legend_guide.html#creating-artists-specifically-for-add
```




```
y_pred=model.predict(X_test)
print(y_pred)
```

```
[[7.8544663e-06 5.1280385e-04 9.1374549e-04 ... 6.7149056e-04
  6.2206638e-04 1.1135340e-03]
 [5.1297643e-04 5.6695109e-03 1.6272863e-02 ... 2.2483338e-02
  6.0521476e-03 1.5810097e-02]
 [1.4559590e-04 3.5141567e-03 2.9652247e-03 ... 2.3180703e-03
  2.6174374e-03 1.1853079e-03]
 ...
 [3.2761672e-03 1.0035246e-02 8.7864920e-03 ... 1.0732072e-02
  8.9350538e-03 7.5503672e-03]
 [1.5082180e-03 4.8265620e-03 1.4429620e-02 ... 1.7857015e-02
  7.5507029e-03 1.3130407e-02]
 [7.4868605e-02 5.3141542e-02 2.4259845e-03 ... 4.5230179e-04
  9.6727135e-03 8.7665208e-03]]
```

✓ 15 giây hoàn thành lúc 21:39