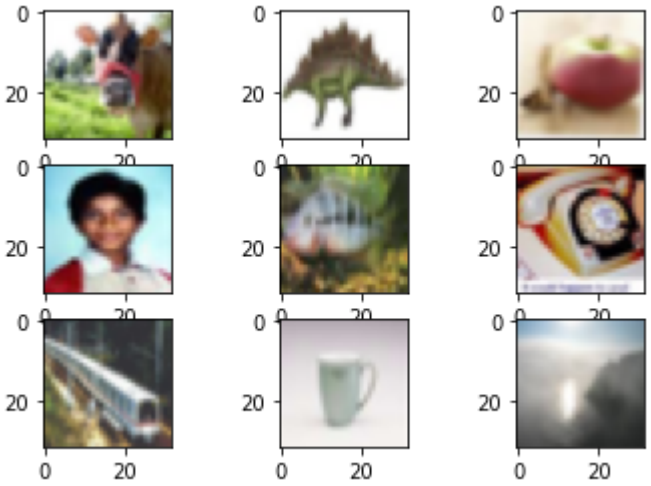
```
from tensorflow.keras.optimizers import Adam

import numpy as np
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.utils import np_utils
from keras.models import Sequential
from keras.datasets import cifar100
(x_train,y_train),(x_test,y_test)= cifar100.load_data()
```

```
import matplotlib.pyplot as plt
for i in range (9):
  plt.subplot(330+i+1)
  plt.imshow(x_train[i])

plt.show()
```



```
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

```
((50000, 32, 32, 3), (50000, 1), (10000, 32, 32, 3), (10000, 1))
```

Nhấp đúp (hoặc nhấn Enter) để chỉnh sửa

```
from tensorflow.keras.utils import to_categorical

x_train= x_train.astype('float32')
x_test= x_test.astype('float32')
x_train/=255
```

```
x_test/=255
y_train= to_categorical (y_train,100)
y_test= to_categorical (y_test,100)

from keras.layers import Dense
from keras.layers.convolutional import MaxPooling2D
from keras.layers import Flatten
from tensorflow.keras.layers import Conv2D
model = Sequential()

model.add(Conv2D(32,(3,3),input_shape=(32,32,3),padding='same',activation='relu'))
model.add(Dropout(0.2))

model.add(Conv2D(32,(3,3),activation='relu',padding='same'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(64,(3,3),activation='relu',padding='same'))
model.add(Dropout(0.2))

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

model.add(Conv2D(128,(3,3),activation='relu',padding='same'))
model.add(Dropout(0.2))

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

model.add(Flatten())
model.add(Dropout(0.2))

model.add(Dense(1024,activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(512,activation='relu'))
model.add(Dropout(0.2))



model.add(Dense(100,activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 dropout (Dropout)           (None, 32, 32, 32)        0

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        9248

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )

 conv2d_2 (Conv2D)           (None, 16, 16, 64)        18496
```

```
 dropout_1 (Dropout)         (None, 16, 16, 64)        0

 conv2d_3 (Conv2D)           (None, 16, 16, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 conv2d_4 (Conv2D)           (None, 8, 8, 128)         73856

 dropout_2 (Dropout)         (None, 8, 8, 128)         0

 conv2d_5 (Conv2D)           (None, 8, 8, 128)         147584

 max_pooling2d_2 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 flatten (Flatten)           (None, 2048)              0

 dropout_3 (Dropout)         (None, 2048)              0

 dense (Dense)               (None, 1024)              2098176

 dropout_4 (Dropout)         (None, 1024)              0

 dense_1 (Dense)             (None, 512)               524800

 dropout_5 (Dropout)         (None, 512)               0

 dense_2 (Dense)             (None, 100)               51300

=================================================================
Total params: 2,961,284
Trainable params: 2,961,284
Non-trainable params: 0
_________________________________________________________________
```

```
from tensorflow.keras.optimizers import SGD
#opt = SGD(lr = 0.0005, momentum= 0.9) #lr la toc do hoc, momentum la dong luong
model.compile(optimizer=Adam(learning_rate=0.0005), loss='categorical_crossentropy', metri

history=model.fit(x_train,
                  y_train,
                  epochs=20,
                  batch_size=64,
                  verbose=1,
                  validation_data=(x_test,y_test))
```

```
Epoch 1/20
782/782 [==============================] - 9s 11ms/step - loss: 0.8534 - accuracy: 0
Epoch 2/20
782/782 [==============================] - 8s 10ms/step - loss: 0.8064 - accuracy: 0
Epoch 3/20
782/782 [==============================] - 8s 10ms/step - loss: 0.7757 - accuracy: 0
Epoch 4/20
782/782 [==============================] - 8s 10ms/step - loss: 0.7434 - accuracy: 0
Epoch 5/20
782/782 [==============================] - 8s 10ms/step - loss: 0.7125 - accuracy: 0
```

```
Epoch 6/20
782/782 [==============================] - 8s 10ms/step - loss: 0.6955 - accuracy: 0
Epoch 7/20
782/782 [==============================] - 8s 10ms/step - loss: 0.6676 - accuracy: 0
Epoch 8/20
782/782 [==============================] - 8s 10ms/step - loss: 0.6555 - accuracy: 0
Epoch 9/20
782/782 [==============================] - 8s 10ms/step - loss: 0.6323 - accuracy: 0
Epoch 10/20
782/782 [==============================] - 8s 10ms/step - loss: 0.6144 - accuracy: 0
Epoch 11/20
782/782 [==============================] - 8s 10ms/step - loss: 0.5922 - accuracy: 0
Epoch 12/20
782/782 [==============================] - 8s 10ms/step - loss: 0.5850 - accuracy: 0
Epoch 13/20
782/782 [==============================] - 8s 10ms/step - loss: 0.5633 - accuracy: 0
Epoch 14/20
782/782 [==============================] - 8s 10ms/step - loss: 0.5495 - accuracy: 0
Epoch 15/20
782/782 [==============================] - 8s 10ms/step - loss: 0.5461 - accuracy: 0
Epoch 16/20
782/782 [==============================] - 8s 10ms/step - loss: 0.5247 - accuracy: 0
Epoch 17/20
782/782 [==============================] - 8s 10ms/step - loss: 0.5169 - accuracy: 0
Epoch 18/20
782/782 [==============================] - 8s 10ms/step - loss: 0.5102 - accuracy: 0
Epoch 19/20
782/782 [==============================] - 8s 10ms/step - loss: 0.4998 - accuracy: 0
Epoch 20/20
782/782 [==============================] - 8s 10ms/step - loss: 0.4959 - accuracy: 0
```

```
model.save("dulieucifar100.h5")
```

```
from keras.models import load_model
new_model = load_model ('dulieucifar100.h5')
```

```
from keras.preprocessing.image import load_img,img_to_array
img=load_img('/content/tiger.jpg',target_size=(32,32))
plt.imshow(img)
img=img_to_array(img)
img=img.reshape(1,32,32,3)
img=img.astype('float32')
img=img/255
img.shape
```

```
(1, 32, 32, 3)
```


```
import numpy as np
np.argmax(new_model.predict(img),axis=1)
```

```
array([88])
```