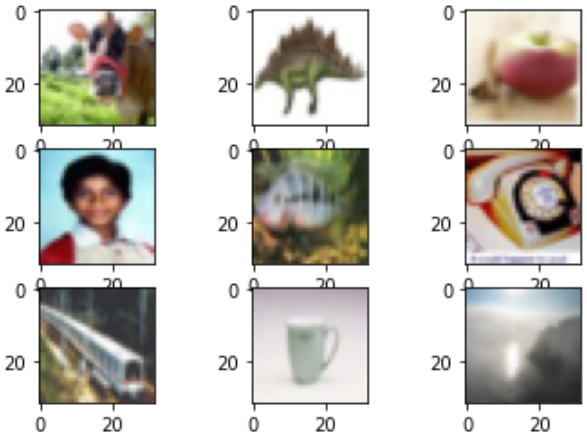
```
from tensorflow.keras.optimizers import Adam

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()
```

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



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

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

```
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', metrics=['accuracy'

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 [==============================] - 20s 11ms/step - loss: 4.1400 - accuracy: 0.0609 - val_l
Epoch 2/20
782/782 [==============================] - 8s 10ms/step - loss: 3.4455 - accuracy: 0.1724 - val_lo
Epoch 3/20
782/782 [==============================] - 8s 10ms/step - loss: 3.0414 - accuracy: 0.2467 - val_lo
Epoch 4/20
782/782 [==============================] - 8s 11ms/step - loss: 2.7626 - accuracy: 0.3032 - val_lo
Epoch 5/20
782/782 [==============================] - 8s 11ms/step - loss: 2.5347 - accuracy: 0.3509 - val_lo
Epoch 6/20
782/782 [==============================] - 8s 10ms/step - loss: 2.3338 - accuracy: 0.3905 - val_lo
Epoch 7/20
782/782 [==============================] - 10s 12ms/step - loss: 2.1638 - accuracy: 0.4280 - val_l
Epoch 8/20
782/782 [==============================] - 10s 12ms/step - loss: 1.9995 - accuracy: 0.4626 - val_l
Epoch 9/20
782/782 [==============================] - 9s 11ms/step - loss: 1.8543 - accuracy: 0.4967 - val_lo
Epoch 10/20
782/782 [==============================] - 8s 10ms/step - loss: 1.7227 - accuracy: 0.5264 - val_lo
Epoch 11/20
782/782 [==============================] - 8s 11ms/step - loss: 1.5980 - accuracy: 0.5525 - val_lo
Epoch 12/20
782/782 [==============================] - 8s 10ms/step - loss: 1.4878 - accuracy: 0.5808 - val_lo
Epoch 13/20
782/782 [==============================] - 8s 10ms/step - loss: 1.3886 - accuracy: 0.6038 - val_lo
```

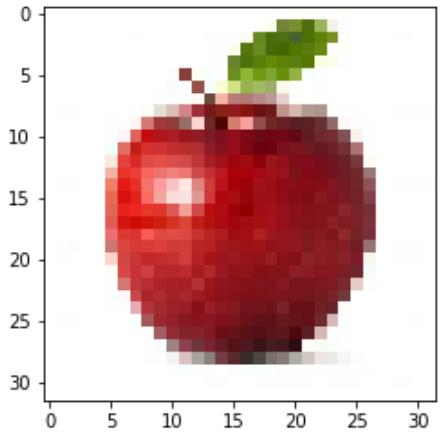
```
Epoch 14/20
782/782 [==============================] - 8s 10ms/step - loss: 1.2961 - accuracy: 0.6249 - val_lo
Epoch 15/20
782/782 [==============================] - 8s 10ms/step - loss: 1.2120 - accuracy: 0.6472 - val_lo
Epoch 16/20
782/782 [==============================] - 8s 10ms/step - loss: 1.1460 - accuracy: 0.6620 - val_lo
Epoch 17/20
782/782 [==============================] - 8s 10ms/step - loss: 1.0839 - accuracy: 0.6802 - val_lo
Epoch 18/20
782/782 [==============================] - 8s 10ms/step - loss: 1.0176 - accuracy: 0.6951 - val_lo
Epoch 19/20
782/782 [==============================] - 8s 10ms/step - loss: 0.9655 - accuracy: 0.7118 - val_lo
Epoch 20/20
782/782 [==============================] - 8s 10ms/step - loss: 0.9199 - accuracy: 0.7205 - val_lo
```

```
model.save("cifar100.h5")
```

```
from keras.models import load_model
model5 = load_model('cifar100.h5')
```

```
from keras.preprocessing.image import load_img,img_to_array
img=load_img('/content/images (5).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(model5.predict(img),axis=1)
```

```
array([0])
```