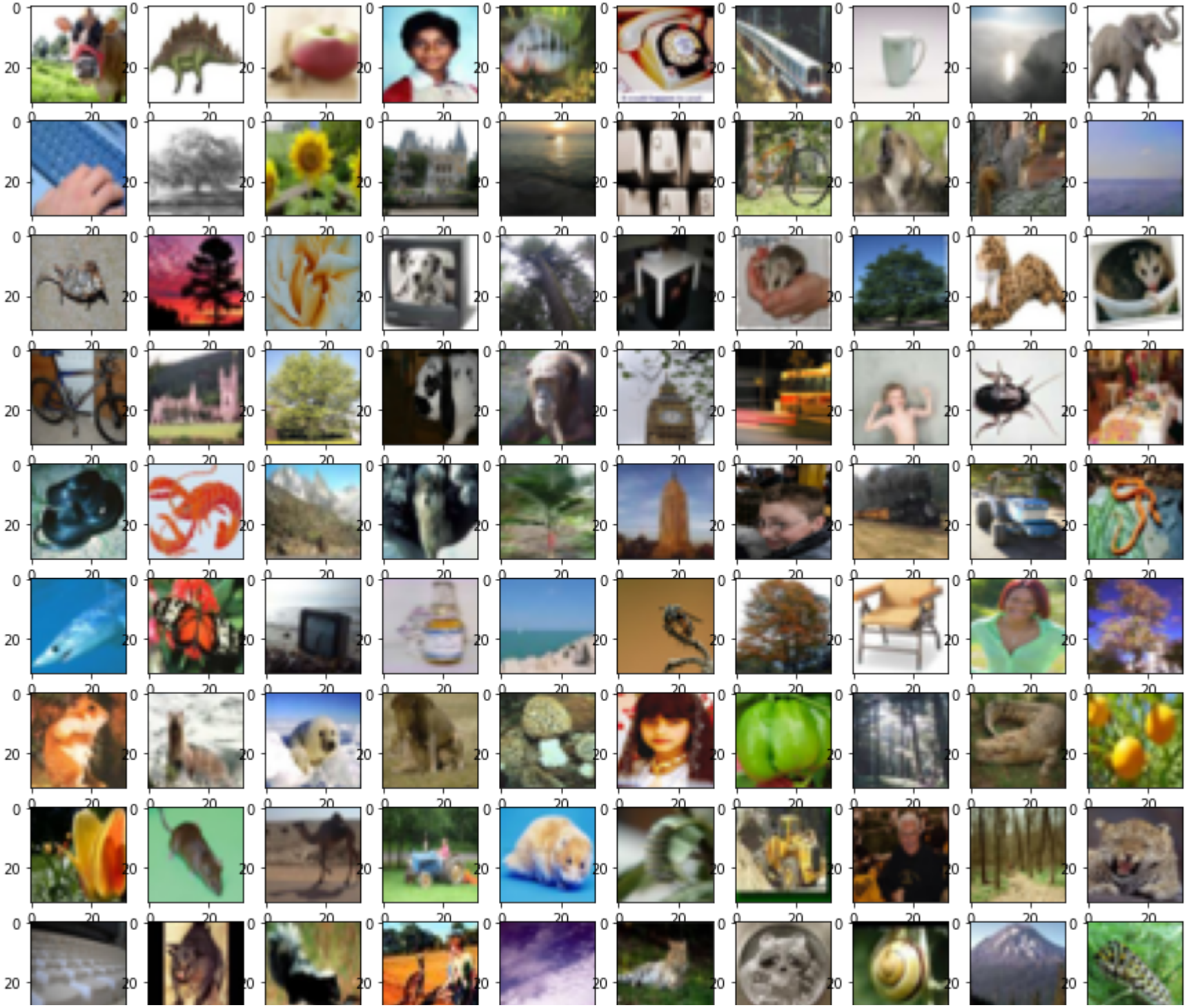
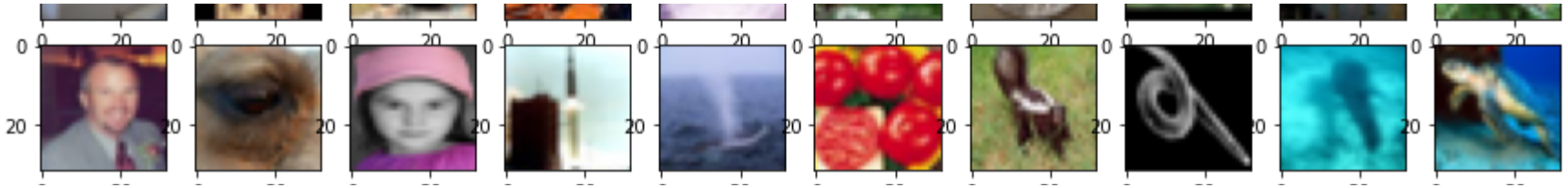
```
from keras.datasets import cifar10
from keras.datasets import cifar100
from matplotlib import pyplot as plt
import numpy as np
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.utils import np_utils
from keras.layers import Dense,Activation,Dropout,LSTM,BatchNormalization
from keras.layers import Flatten
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.utils import load_img,img_to_array
from tensorflow.keras.models import load_model
from tensorflow.keras.optimizers import RMSprop
```

```
from keras.datasets import cifar10
from keras.datasets import cifar100
import matplotlib.pyplot as plt
from tensorflow.keras.utils import to_categorical
(x_train,y_train),(x_test,y_test) = cifar100.load_data()
x_train.shape
y_train.shape
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz
169009152/169001437 [==============================] - 2s 0us/step
169017344/169001437 [==============================] - 2s 0us/step
(50000, 1)
```

```
from numpy import subtract
plt.figure(figsize=(15,15))
for i in range (100):
  plt.subplot(10,10,i+1)
  plt.imshow(x_train[i])
plt.show()
```





```
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)
```

```
x_test.shape
```

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

```
x_train.shape
```

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

```
y_test.shape
```

```
(10000, 100)
```

```
y_train.shape
```

```
(50000, 100)
```

```
model = Sequential()
model.add(Flatten(input_shape=(32,32,3)))
model.add(Dense(784,activation='relu'))
model.add(Dense(784,activation='relu'))
model.add(Dense(100,activation='Softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 3072)              0

 dense (Dense)               (None, 784)               2409232

 dense_1 (Dense)             (None, 784)               615440

 dense_2 (Dense)             (None, 100)               78500

=================================================================
Total params: 3,103,172
Trainable params: 3,103,172
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(loss='categorical_crossentropy',optimizer=RMSprop(), metrics=['accuracy'])
history=model.fit(x_train,y_train,batch_size=128,epochs=50,verbose=1,validation_data=(x_test,y_test))
```

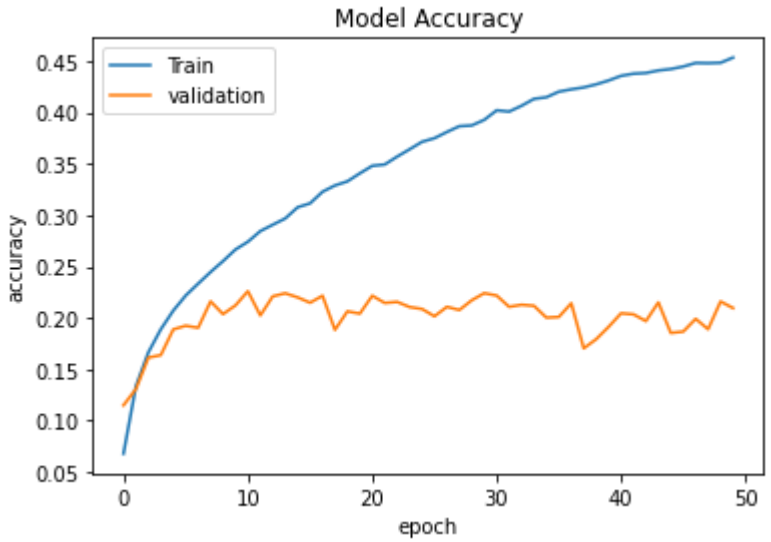
```
Epoch 1/50
391/391 [==============================] - 24s 59ms/step - loss: 4.2329 - accuracy: 0.0679 - val_loss: 3.8426 - val_
Epoch 2/50
391/391 [==============================] - 22s 57ms/step - loss: 3.7276 - accuracy: 0.1336 - val_loss: 3.7177 - val_
Epoch 3/50
391/391 [==============================] - 26s 67ms/step - loss: 3.5396 - accuracy: 0.1665 - val_loss: 3.5838 - val_
Epoch 4/50
391/391 [==============================] - 22s 57ms/step - loss: 3.4179 - accuracy: 0.1890 - val_loss: 3.5784 - val_
Epoch 5/50
391/391 [==============================] - 25s 63ms/step - loss: 3.3162 - accuracy: 0.2071 - val_loss: 3.5018 - val_
Epoch 6/50
391/391 [==============================] - 26s 66ms/step - loss: 3.2375 - accuracy: 0.2218 - val_loss: 3.4675 - val_
Epoch 7/50
391/391 [==============================] - 24s 60ms/step - loss: 3.1673 - accuracy: 0.2334 - val_loss: 3.4642 - val_
Epoch 8/50
391/391 [==============================] - 23s 58ms/step - loss: 3.1074 - accuracy: 0.2447 - val_loss: 3.4146 - val_
Epoch 9/50
391/391 [==============================] - 22s 57ms/step - loss: 3.0620 - accuracy: 0.2553 - val_loss: 3.4894 - val_
Epoch 10/50
391/391 [==============================] - 23s 59ms/step - loss: 3.0037 - accuracy: 0.2666 - val_loss: 3.4431 - val_
```

```
Epoch 11/50
391/391 [==============================] - 22s 57ms/step - loss: 2.9598 - accuracy: 0.2741 - val_loss: 3.4031 - val_
Epoch 12/50
391/391 [==============================] - 22s 57ms/step - loss: 2.9102 - accuracy: 0.2845 - val_loss: 3.5137 - val_
Epoch 13/50
391/391 [==============================] - 22s 57ms/step - loss: 2.8696 - accuracy: 0.2907 - val_loss: 3.5092 - val_
Epoch 14/50
391/391 [==============================] - 22s 57ms/step - loss: 2.8360 - accuracy: 0.2968 - val_loss: 3.5661 - val_
Epoch 15/50
391/391 [==============================] - 22s 57ms/step - loss: 2.7963 - accuracy: 0.3077 - val_loss: 3.6432 - val_
Epoch 16/50
391/391 [==============================] - 22s 58ms/step - loss: 2.7606 - accuracy: 0.3115 - val_loss: 3.6338 - val_
Epoch 17/50
391/391 [==============================] - 22s 57ms/step - loss: 2.7216 - accuracy: 0.3228 - val_loss: 3.6912 - val_
Epoch 18/50
391/391 [==============================] - 22s 57ms/step - loss: 2.6901 - accuracy: 0.3289 - val_loss: 3.6787 - val_
Epoch 19/50
391/391 [==============================] - 22s 57ms/step - loss: 2.6689 - accuracy: 0.3329 - val_loss: 3.8737 - val_
Epoch 20/50
391/391 [==============================] - 22s 57ms/step - loss: 2.6340 - accuracy: 0.3407 - val_loss: 3.9317 - val_
Epoch 21/50
391/391 [==============================] - 22s 57ms/step - loss: 2.6096 - accuracy: 0.3480 - val_loss: 3.7702 - val_
Epoch 22/50
391/391 [==============================] - 22s 57ms/step - loss: 2.5916 - accuracy: 0.3493 - val_loss: 3.7843 - val_
Epoch 23/50
391/391 [==============================] - 22s 57ms/step - loss: 2.5633 - accuracy: 0.3570 - val_loss: 3.9026 - val_
Epoch 24/50
391/391 [==============================] - 22s 57ms/step - loss: 2.5256 - accuracy: 0.3641 - val_loss: 3.9422 - val_
Epoch 25/50
391/391 [==============================] - 22s 57ms/step - loss: 2.5054 - accuracy: 0.3714 - val_loss: 3.9097 - val_
Epoch 26/50
391/391 [==============================] - 22s 57ms/step - loss: 2.4862 - accuracy: 0.3749 - val_loss: 3.9967 - val_
Epoch 27/50
391/391 [==============================] - 22s 57ms/step - loss: 2.4626 - accuracy: 0.3810 - val_loss: 4.3639 - val_
Epoch 28/50
391/391 [==============================] - 22s 57ms/step - loss: 2.4402 - accuracy: 0.3866 - val_loss: 4.2445 - val_
Epoch 29/50
```

```
score = model.evaluate(x_test,y_test,verbose=0)
print('Sai số kiểm tra là: ',score[0])
print('Độ chính xác kiểm tra là: ',score[1])
```

```
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()
```

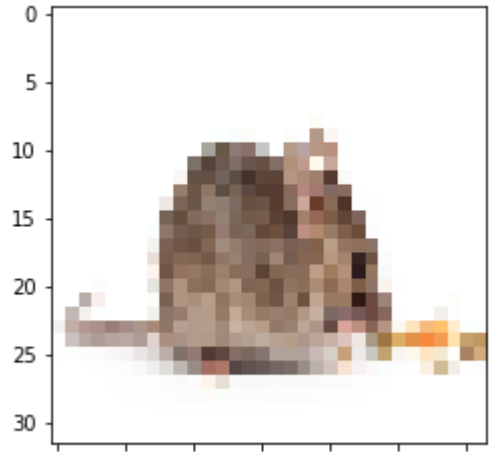
```
Sai số kiểm tra là:  5.627303600311279
Độ chính xác kiểm tra là:  0.20960000157356262
```




```
model.save('/content/drive/MyDrive/BT AI/cifar100.h5')
```

```
img = load_img('/content/drive/MyDrive/Anh test/chuot1.jpg',target_size=(32,32))
plt.imshow(img)
img = img_to_array(img)
img = img.reshape(1,32,32,3)
img = img/255
np.argmax(model.predict(img),axis=-1)
```

```
array([29])
```




```
from google.colab import drive
drive.mount('/content/drive')
```