THỰC HÀNH 3 : ARTIFICIAL NEURAL NETWORK

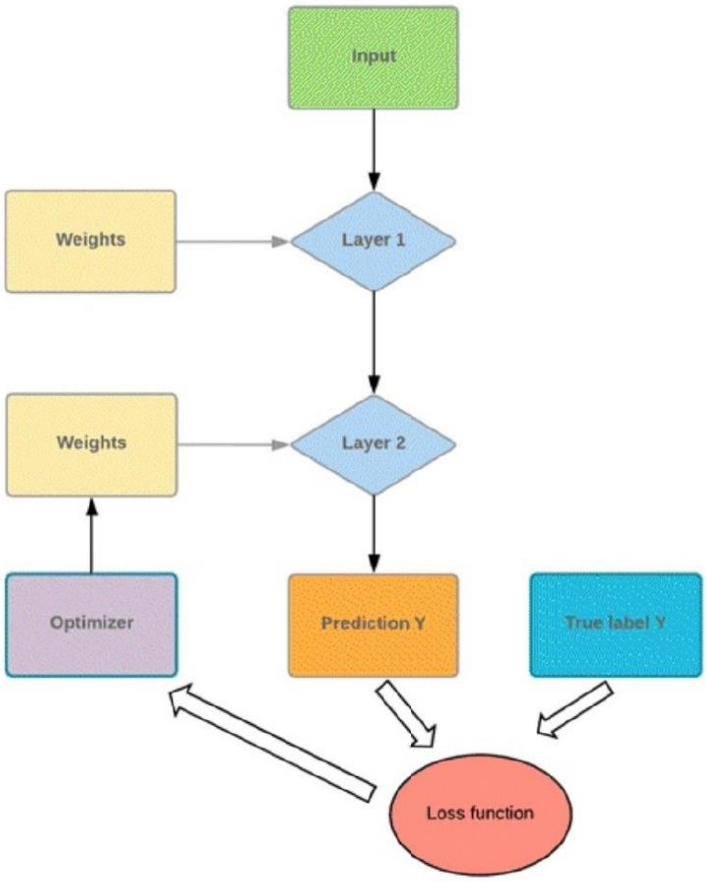
Sau bài thực hành này, sinh viên có khả năng:

- Xây dựng được kiến trúc Neural Network
- Xây dựng mô hình Neural Network bằng tensorflow
- Cài đặt ứng dụng Neural Network bằng tensorflow

1. GIỚI THIỆU

Một Artificial Neural Network (ANN) gồm 4 đối tượng sau

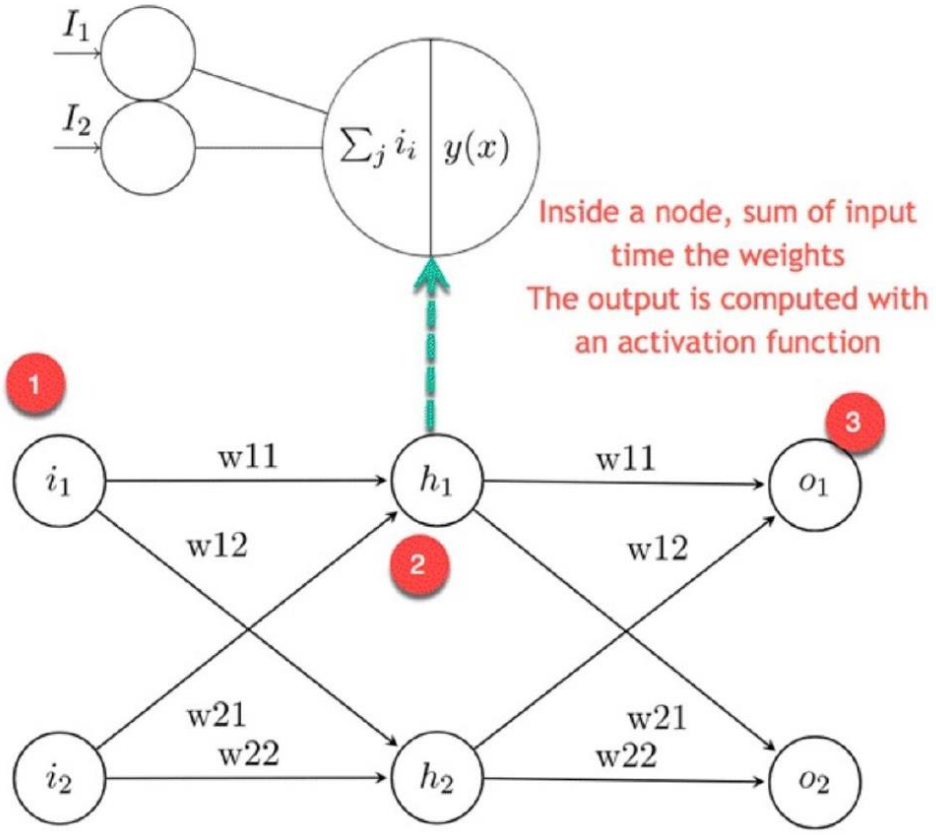
- **Layers**: tất cả quá trình học xảy ra trong các layer này gồm : Input, Hidden và Output
- **Feature và label**: dữ liệu đầu vào của ANN (feature) và đầu ra từ ANN (label)
- **Loss function**: đo lường hiệu quả của việc học
- **Optimizer**: cải tiến quá trình học bằng cách cập nhật tri thức trong mạng



2. KIẾN TRÚC NEURAL NETWORK

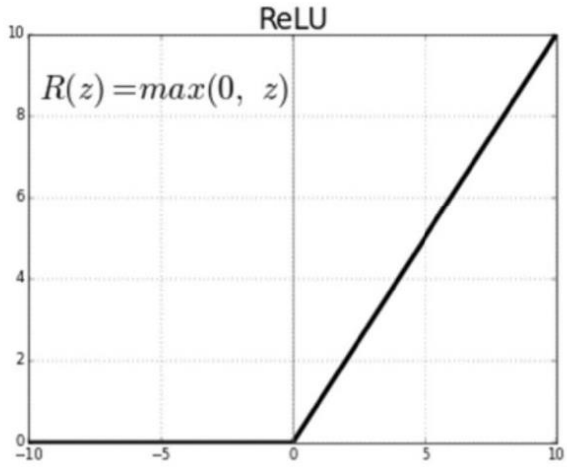
2.1. Layers

- Một layer là nơi diễn ra quá trình học.
- Một layer có nhiều **weight (neuron)**
- Một neural network thường được xử lý bởi tầng fully connected layers.
- Đầu vào một neural network là một vector và một scalar chứa các label.
- Các giá trị đầu vào sẽ được gởi tới các neuron và tính đầu ra với hàm **activation**



2.2. Hàm activation

- Hàm activation của một neuron tính toán đầu ra của tập input.
- Hàm activation giúp neural network học các mẫu phi tuyến tính
- Hàm activation phổ biến là ReLU (Rectified linear unit): gán giá trị 0 cho các giá trị âm



ReLU Activation Function

Ngoài ra, còn một số hàm activation khác

- Piecewise Linear
- Sigmoid
- Tanh
- Leaky Relu

2.3. Hàm Loss

Hàm loss để đánh giá hiệu quả của mô hình khi huấn luyện. Trong quá trình huấn luyện, hàm loss này sẽ cực tiểu hóa

2.4. Hàm Optimizer

Hàm optimizer sẽ cải tiến các weight của ANN để cực tiểu hóa hàm loss. Optimizer phổ biến là Stochastic Gradient Descent

2.5. Hạn chế của neural network

2.5.1. Overfitting

- Một mạng neural phức tạp khó dự báo được dữ liệu chưa biết
- Mạng neural có nhiều weight sẽ dễ dàng nhận biết các dữ liệu trong tập huấn luyện nhưng bị overfit
- Nếu dữ liệu không cân bằng trong nhóm (không đủ dữ liệu trong nhóm), network sẽ học rất tốt trong quá trình huấn luyện nhưng không có khả năng dự báo dữ liệu chưa biết
- Có sự đánh đổi optimization và generalization
- Optimization là tìm những tham số để cực tiểu hóa hàm loss của tập huấn luyện
- Generalization là cho biết mô hình dự báo dữ liệu chưa biết

- Để tránh trường hợp overfitting còn gọi là regualarization, ta cân bằng dữ liệu với khối lượng đủ lớn.

2.5.2. Network size

- NN có quá nhiều tầng và hidden units rất phức tạp.
- Một cách giảm độ phức tạp của mô hình là giảm kích thước NN
- Không có quy tắc nào xác định trước số tầng.
- Chúng ta bắt đầu từ số tầng nhỏ và tăng kích thước NN cho đến khi overfit.

2.5.3. Weight Regularization

Một kỹ thuật để ngăn overfit là thêm ràng buộc (contraint) tới weight của NN. Constraint bắt buộc kích thước của NN chỉ nhận giá trị nhỏ. Constraint được thêm vào error của hàm loss. Có 2 loại

L1: Lasso: Chi phí tỉ lệ thuận với giá trị tuyệt đối của hệ số weight

L2: Ridge: Chi phí tỉ lệ thuận với bình phương giá trị của hệ số weight

2.5.4. Dropout

- NN sử dụng dropout nghĩa là vài weight sẽ ngẫu nhiên gán giá trị 0. Ví dụ, ta có mảng weight [0.1, 1.7, 0.7, -0.9].
- Dropout sẽ ngẫu nhiên gán 0 và kết quả là [0.1, 0, 0, -0.9]. Tham số điều khiển dropout là dropout rate.
- Rate định nghĩa có bao nhiêu weight được gán giá trị 0. Thông thường rate nằm trong [0.2, 0.5]

3. CÀI ĐẶT ARTIFICIAL NEURAL NETWORK CƠ BẢN

Phần này sẽ hướng dẫn cài đặt ANN cho bài toán phân lớp.

Đầu vào:

- Tập ảnh chữ viết tay kích thước 28 x 28

Đầu ra:

- một mảng chứa xác suất các label (mang giá trị 0 đến 9)

3.1. Nạp thư viện

```
[6]  #install required libraries
     import pandas as pd
     import numpy as np
     #data visualization packages
     import matplotlib.pyplot as plt
     #keras packages
     import tensorflow as tf
     import keras
     from keras.models import Sequential
     from keras.layers import Convolution2D
     from keras.layers import MaxPooling2D
     from keras.layers import Flatten
     from keras.layers import Dense
     from keras.wrappers.scikit_learn import KerasClassifier
     from keras.layers import Dropout
     #model evaluation packages
     from sklearn.metrics import f1_score, roc_auc_score, log_loss
     from sklearn.model_selection import cross_val_score, cross_validate
     from sklearn.preprocessing import MinMaxScaler
```

3.2. Đọc tập dữ liệu FASHION MNIST

Tập dataset Fashion MNIST là tập hợp ảnh thời trang của Zalando kích thước 28 x 28. Tập này có 60,000 mẫu train và 10,000 mẫu test. Có 10 nhãn hỗ trợ cho việc phân lớp từng mẫu.

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

```
[7]  #tải bộ mnist fashion dataset
     fashion = tf.keras.datasets.fashion_mnist
     (X_train, y_train), (X_test, y_test) = fashion.load_data()
     print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)

     Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
     29515/29515 [==============================] - 0s 0us/step
     Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
     26421880/26421880 [==============================] - 0s 0us/step
     Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
     5148/5148 [==============================] - 0s 0us/step
     Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
     4422102/4422102 [==============================] - 0s 0us/step
     (60000, 28, 28) (60000,) (10000, 28, 28) (10000,)
```

3.3. Xử lý dữ liệu

Đây là bước quan trọng trong quá trình xây dựng model. Gồm các bước như sau

- Xử lý dữ liệu thiếu
- Biểu diễn đặc trưng
- Reshape dữ liệu
- Mã hóa label thành One-Hot encoding
- Tách dataset thành training(80%) và testing (20%)

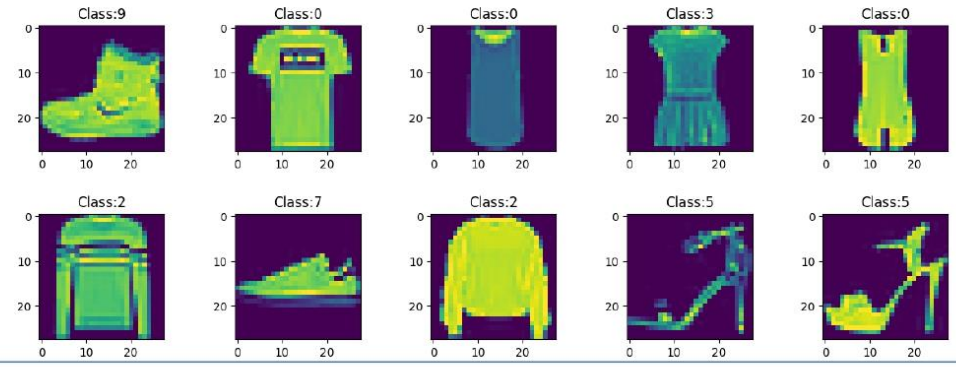
```
#reshape data from 3-D to 2-D array
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)

#feature scaling
minmax = MinMaxScaler()

X_train = minmax.fit_transform(X_train)
X_test = minmax.transform(X_test)

print('Number of classes:', len(np.unique(y_train)))
print('Classes:', np.unique(y_train))
```

```
Number of classes: 10
Classes: [0 1 2 3 4 5 6 7 8 9]
```

3.4. Trực quan hóa dữ liệu

Ta cần trực quan hóa dữ liệu trước khi xây dựng mô hình. Trực quan hóa giúp ta phân biệt dữ liệu.

```
#Data visualization
fig, axes = plt.subplots(nrows=2, ncols=5,figsize=(15, 5))
ax = axes.ravel()
for i in range(10):
  ax[i].imshow(X_train[i].reshape(28, 28))
  ax[i].title.set_text('Class:'+str(y_train[i]))
plt.subplots_adjust(hspace=0.5)
plt.show()
```



3.5. Xây dựng mô hình học ANN

```
[11] #initial ANN model
     fashion_model = Sequential()

     #add first hidden layer
     fashion_model.add(Dense(input_dim=X_train.shape[1], units=256, kernel_initializer='uniform', activation='relu'))

     #add output layer
     fashion_model.add(Dense(units=10, kernel_initializer='uniform', activation='softmax'))

     #compile neural network
     fashion_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

     #model summary
     fashion_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 256)               200960

 dense_1 (Dense)             (None, 10)                2570

=================================================================
Total params: 203,530
Trainable params: 203,530
Non-trainable params: 0
_________________________________________________________________
```

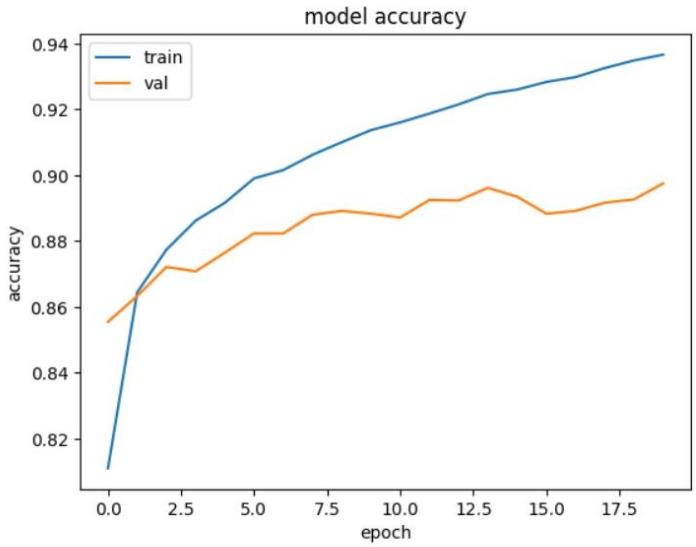
3.6. Huấn luyện ANN model

```
[59] model_fit = fashion_model.fit(X_train, y_train, validation_split=0.1, epochs=20, verbose=1)

     Epoch 1/20
     1688/1688 [==============================] - 17s 9ms/step - loss: 0.5165 - accuracy: 0.8111 - val_loss: 0.4035 - val_accuracy: 0.8555
     Epoch 2/20
     1688/1688 [==============================] - 16s 10ms/step - loss: 0.3723 - accuracy: 0.8645 - val_loss: 0.3690 - val_accuracy: 0.8633
     Epoch 3/20
     1688/1688 [==============================] - 15s 9ms/step - loss: 0.3350 - accuracy: 0.8774 - val_loss: 0.3643 - val_accuracy: 0.8722
     Epoch 4/20
     1688/1688 [==============================] - 15s 9ms/step - loss: 0.3063 - accuracy: 0.8862 - val_loss: 0.3696 - val_accuracy: 0.8708
     Epoch 5/20
     1688/1688 [==============================] - 16s 9ms/step - loss: 0.2884 - accuracy: 0.8916 - val_loss: 0.3547 - val_accuracy: 0.8765
     Epoch 6/20
     1688/1688 [==============================] - 18s 10ms/step - loss: 0.2737 - accuracy: 0.8990 - val_loss: 0.3357 - val_accuracy: 0.8823
     Epoch 7/20
```

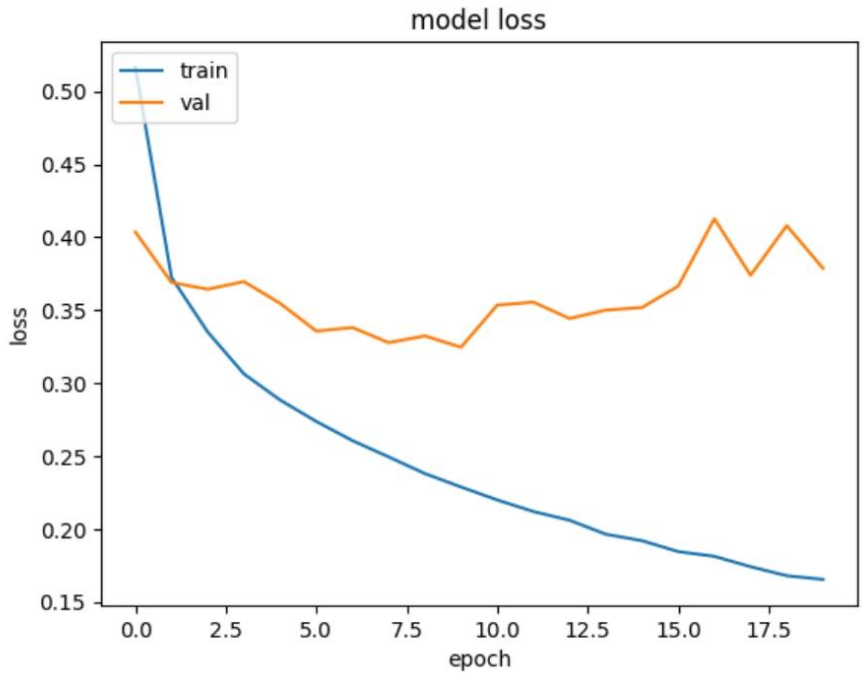
```
Epoch 17/20
1688/1688 [==============================] - 17s 10ms/step - loss: 0.1813 - accuracy: 0.9298 - val_loss: 0.4127 - val_accuracy: 0.8892
Epoch 18/20
1688/1688 [==============================] - 17s 10ms/step - loss: 0.1742 - accuracy: 0.9325 - val_loss: 0.3739 - val_accuracy: 0.8917
Epoch 19/20
1688/1688 [==============================] - 17s 10ms/step - loss: 0.1680 - accuracy: 0.9348 - val_loss: 0.4079 - val_accuracy: 0.8927
Epoch 20/20
1688/1688 [==============================] - 16s 10ms/step - loss: 0.1655 - accuracy: 0.9366 - val_loss: 0.3787 - val_accuracy: 0.8975
```

3.7. Đánh giá ANN model

```
[60] plt.plot(model_fit.history['accuracy'])
     plt.plot(model_fit.history['val_accuracy'])
     plt.title('model accuracy')
     plt.ylabel('accuracy')
     plt.xlabel('epoch')
     plt.legend(['train', 'val'], loc='upper left')
     plt.show()
```



```
plt.plot(model_fit.history['loss'])
plt.plot(model_fit.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



3.8. Dự báo ảnh mới

```
print(np.argmax(fashion_model.predict(X_test)[0]), y_test[0])
print(np.argmax(fashion_model.predict(X_test)[10]), y_test[10])
```

```
313/313 [==============================] - 1s 2ms/step
9 9
313/313 [==============================] - 1s 3ms/step
4 4
```

3.9. Đọc tập dữ liệu từ file

```
[4] mnist_train_path = '/content/sample_data/fashion-mnist_train.csv'
    mnist_test_path = '/content/sample_data/fashion-mnist_test.csv'
    mnist_train = pd.read_csv(mnist_train_path)
    mnist_test = pd.read_csv(mnist_test_path)
    print(mnist_train.head())
    print(mnist_train.shape)
```

```
X_train = mnist_train.iloc[:,1:]
X_test = mnist_test.iloc[:,1:]
print(X_train.shape)

y_train = mnist_train.iloc[:,0]
y_test = mnist_test.iloc[:,0]

print('Number of classes:', len(np.unique(y_train)))
print('Classes:', np.unique(y_train))
```

```
(60000, 784)
Number of classes: 10
Classes: [0 1 2 3 4 5 6 7 8 9]
```

```
[10] import tensorflow as tf
     import keras
     from keras import Sequential
     from keras.layers import Dense
```

```
[15] model = Sequential()
     model.add(Dense(input_dim=X_train.shape[1], units=256, kernel_initializer='uniform', activation='relu'))
     model.add(Dense(units=10, kernel_initializer='uniform', activation='softmax'))
     model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'] )
     model.summary()
```

```
[20] model_fit = model.fit(X_train, y_train, epochs=30, verbose=1)
```

```
print(np.argmax(model.predict(X_test)[0]), y_test[0])
```

```
img_size = (28, 28)
img = tf.keras.preprocessing.image.load_img(
   '/content/sample_data/dress.jpg', color_mode='grayscale', target_size=img_size)

img_array = tf.keras.preprocessing.image.img_to_array(img)
img_array = tf.squeeze(img_array)
img = np.reshape(img_array,(784))
img_array = tf.expand_dims(img,0)
img_array = tf.convert_to_tensor(img_array)
print(img_array.shape)
#print(img_array.reshape(None, 784))

img_predict = model.predict(img_array)
print(img_predict)
score = np.argmax(img_predict)
print(score)
```

4. BÀI TẬP

1. Viết chương trình cài đặt ANN để nhận dạng ảnh trên bộ dataset CIFAR10 có sẵn trong tensorflow với các nhãn sau

Label	Description
0	airplane
1	automobile
2	bird
3	cat
4	deer
5	dog
6	frog
7	horse
8	ship
9	truck

2. Viết chương trình cài đặt ANN để nhận dạng ảnh chữ viết số trên bộ dataset MNIST do giảng viên cung cấp với các nhãn sau

Label	Nhãn
0	Số 0
1	Số 1
2	Số 2
3	Số 3
4	Số 4
5	Số 5

6	Số 6
7	Số 7
8	Số 8
9	Số 9

3. Cho đoạn mã đọc ảnh của Cat và Dog

```
# load dogs vs cats dataset, reshape and save to a new file
from os import listdir
from numpy import asarray
from numpy import save
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
# define location of dataset
folder = 'train/'
photos, labels = list(), list()
# enumerate files in the directory
for file in listdir(folder):
 # determine class
 output = 0.0
 if file.startswith('dog'):
 output = 1.0
 # load image
 photo = load_img(folder + file, target_size=(200, 200))
 # convert to numpy array
 photo = img_to_array(photo)
 # store
 photos.append(photo)
 labels.append(output)
# convert to a numpy arrays
photos = asarray(photos)
labels = asarray(labels)
print(photos.shape, labels.shape)
# save the reshaped photos
save('dogs_vs_cats_photos.npy', photos)
save('dogs_vs_cats_labels.npy', labels)
```

Hãy viết chương trình cài đặt ANN để nhận dạng ảnh Cat hoặc Dog. Dữ liệu do giảng viên cung cấp

4. Viết chương trình cài đặt ANN để dự báo thu nhập một người > 50K/năm hay <= 50K/năm (nhãn cuối) theo bộ dữ liệu adult do giảng viên cung cấp

5. Viết chương trình cài đặt ANN để dự báo việc đánh giá chất lượng xe ô tô (nhãn cuối) từ bộ dữ liệu car do giảng viên cung cấp.

6. Triển khai các câu 1-5 trên nền tảng Web sử dụng Flask