**Giải thích file label\_image.py**

|  |
| --- |
| import argparse  import numpy as np  import tensorflow as tf  tf.compat.v1.disable\_eager\_execution()  def load\_graph(*model\_file*):    graph = tf.Graph()    graph\_def = tf.compat.v1.GraphDef()    with open(*model\_file*, "rb") as f:      graph\_def.ParseFromString(f.read())    with graph.as\_default():      tf.import\_graph\_def(graph\_def)    return graph  def read\_tensor\_from\_image\_file(*file\_name*,  *input\_height*=299,  *input\_width*=299,  *input\_mean*=0,  *input\_std*=255):    input\_name  = "file\_reader"    output\_name = "normalized"    file\_reader = tf.io.read\_file(*file\_name*, input\_name)    if *file\_name*.endswith(".png")   : image\_reader = tf.io.decode\_png(file\_reader, *channels*=3, *name*="png\_reader")    elif *file\_name*.endswith(".gif") : image\_reader = tf.squeeze(tf.io.decode\_gif(file\_reader, *name*="gif\_reader"))    elif *file\_name*.endswith(".bmp") : image\_reader = tf.io.decode\_bmp(file\_reader, *name*="bmp\_reader")    else: image\_reader = tf.io.decode\_jpeg( file\_reader, *channels*=3, *name*="jpeg\_reader")      float\_caster  = tf.cast(image\_reader, tf.float32)    dims\_expander = tf.expand\_dims(float\_caster, 0)    resized = tf.compat.v1.image.resize\_bilinear(        dims\_expander, [*input\_height*, *input\_width*]    )    normalized  = tf.divide(tf.subtract(resized, [*input\_mean*]), [*input\_std*])    sess        = tf.compat.v1.Session()    return sess.run(normalized)  def load\_labels(*label\_file*):    proto\_as\_ascii\_lines = tf.io.gfile.GFile(*label\_file*).readlines()    return [l.rstrip() for l in proto\_as\_ascii\_lines]  if \_\_name\_\_ == "\_\_main\_\_":    file\_name     = "tensorflow/examples/label\_image/data/grace\_hopper.jpg"    model\_file    = "tensorflow/examples/label\_image/data/inception\_v3\_2016\_08\_28\_frozen.pb"    label\_file    = "tensorflow/examples/label\_image/data/imagenet\_slim\_labels.txt"    input\_height  = 299    input\_width   = 299    input\_mean    = 0    input\_std     = 255    input\_layer   = "input"    output\_layer  = "InceptionV3/Predictions/Reshape\_1"    parser = argparse.ArgumentParser()    parser.add\_argument("--image"       , *help*="image to be processed")    parser.add\_argument("--graph"       , *help*="graph/model to be executed")    parser.add\_argument("--labels"      , *help*="name of file containing labels")    parser.add\_argument("--input\_height", *type*=int, *help*="input height")    parser.add\_argument("--input\_width" , *type*=int, *help*="input width")    parser.add\_argument("--input\_mean"  , *type*=int, *help*="input mean")    parser.add\_argument("--input\_std"   , *type*=int, *help*="input std")    parser.add\_argument("--input\_layer" , *help*="name of input layer")    parser.add\_argument("--output\_layer", *help*="name of output layer")    args = parser.parse\_args()    if args.graph:        model\_file    = args.graph    if args.image:        file\_name     = args.image    if args.labels:       label\_file    = args.labels    if args.input\_height: input\_height  = args.input\_height    if args.input\_width:  input\_width   = args.input\_width    if args.input\_mean:   input\_mean    = args.input\_mean    if args.input\_std:    input\_std     = args.input\_std    if args.input\_layer:  input\_layer   = args.input\_layer    if args.output\_layer: output\_layer  = args.output\_layer    graph = load\_graph(model\_file)    t = read\_tensor\_from\_image\_file(        file\_name,  *input\_height*=input\_height,  *input\_width*=input\_width,  *input\_mean*=input\_mean,  *input\_std*=input\_std)    input\_name  = "import/" + input\_layer    output\_name = "import/" + output\_layer    input\_operation   = graph.get\_operation\_by\_name(input\_name)    output\_operation  = graph.get\_operation\_by\_name(output\_name)    with tf.compat.v1.Session(*graph*=graph) as sess:      results = sess.run(output\_operation.outputs[0], {          input\_operation.outputs[0]: t      })    results = np.squeeze(results)    top\_k   = results.argsort()[-5:][::-1]    labels  = load\_labels(label\_file)    for i in top\_k:      print(labels[i], results[i]) |

# 1. Import các thư viện

* **argparse**: Thư viện này được sử dụng để xử lý đối số dòng lệnh. Nó giúp chương trình có thể nhận đối số từ dòng lệnh khi được chạy.
* **numpy**: Thư viện chuyên về xử lý mảng và ma trận trong Python. Trong đoạn mã này, nó được sử dụng để làm việc với mảng số học.
* **tensorflow**: Thư viện chủ đạo của đoạn mã, được sử dụng để xây dựng và thực hiện các mô hình học máy. Đối với phiên bản TensorFlow cũ (1.x), **tf.compat.v1.disable\_eager\_execution()** được sử dụng để tắt tính năng eager execution, giúp tương thích với các phiên bản TensorFlow cũ hơn.

# 2. Định nghĩa hàm load\_graph

**load\_graph(model\_file)**: Hàm này nhận đường dẫn đến file mô hình và trả về một đối tượng đồ thị TensorFlow.

|  |
| --- |
| def load\_graph(model\_file):  graph = tf.Graph()  graph\_def = tf.compat.v1.GraphDef() |

* **graph = tf.Graph()**: Tạo một đối tượng đồ thị TensorFlow mới. Đối tượng này sẽ chứa toàn bộ mô hình và các phép toán của nó.
* **graph\_def = tf.compat.v1.GraphDef()**: Tạo một đối tượng GraphDef, đây là định dạng cho một đồ thị TensorFlow. Nó lưu trữ cấu trúc của đồ thị và các trọng số của mô hình.

|  |
| --- |
| with open(model\_file, "rb") as f:  graph\_def.ParseFromString(f.read()) |

* **with open(model\_file, "rb") as f:**: Mở file mô hình trong chế độ đọc nhị phân ("rb" - read binary). with được sử dụng để đảm bảo rằng file sẽ được đóng tự động sau khi kết thúc khối mã.
* **graph\_def.ParseFromString(f.read())**: Đọc nội dung của file mô hình và sử dụng phương thức ParseFromString để chuyển đổi nó thành một đối tượng GraphDef.

|  |
| --- |
| with graph.as\_default():  tf.import\_graph\_def(graph\_def) |

* **with graph.as\_default():**: Đặt đồ thị graph làm đồ thị mặc định. Các phép toán và biến trong đồ thị được tạo sau đó sẽ thuộc đồ thị này.
* **tf.import\_graph\_def(graph\_def):** Nhập đồ thị từ graph\_def vào đồ thị mặc định của TensorFlow (graph). Các phép toán và biến của mô hình được thêm vào đồ thị.

|  |
| --- |
| return graph |

* **return graph**: Trả về đối tượng đồ thị TensorFlow đã được tạo và nhập mô hình. Đối tượng này có thể được thực hiện các phép toán trong mô hình.

# 3. Định nghĩa hàm read\_tensor\_from\_image\_file

Hàm này dùng để đọc 1 file ảnh, xử lý và chuẩn hóa nó để có thể được sử dụng để làm dữ liệu đầu vào cho mô hình.

|  |
| --- |
| def read\_tensor\_from\_image\_file(file\_name,  input\_height=299,  input\_width=299,  input\_mean=0,  input\_std=255): |

**Bước 1: Xác định tên đối tượng và tên kết quả**

|  |
| --- |
| input\_name = "file\_reader"  output\_name = "normalized" |

* **input\_name** : Tên đối tượng đọc file
* **output\_name**: Tên đối tượng chuẩn hóa dữ liệu

**Bước 2: Đọc và Giải mã File ảnh**

|  |
| --- |
| file\_reader = tf.io.read\_file(file\_name, input\_name)  if file\_name.endswith(".png"):  image\_reader = tf.io.decode\_png(file\_reader, channels=3, name="png\_reader")  elif file\_name.endswith(".gif"):  image\_reader = tf.squeeze(tf.io.decode\_gif(file\_reader, name="gif\_reader"))  elif file\_name.endswith(".bmp"):  image\_reader = tf.io.decode\_bmp(file\_reader, name="bmp\_reader")  else:  image\_reader = tf.io.decode\_jpeg(file\_reader, channels=3, name="jpeg\_reader") |

* **file\_reader**: Đọc nội dung của file ảnh bằng cách sử dụng **tf.io.read\_file**.
* **image\_reader**: Giải mã file ảnh dựa trên định dạng của nó. Sử dụng các hàm như **tf.io.decode\_png**, **tf.io.decode\_gif**, **tf.io.decode\_bmp**, hoặc **tf.io.decode\_jpeg** tùy thuộc vào định dạng.

**Bước 3: Chuyển đổi kiểu và mở rộng số chiều**

|  |
| --- |
| float\_caster = tf.cast(image\_reader, tf.float32)  dims\_expander = tf.expand\_dims(float\_caster, 0) |

* **float\_caster**: Chuyển đổi dữ liệu của ảnh từ kiểu integer sang kiểu float bằng tf.cast.
* **dims\_expander**: Mở rộng số chiều của tensor bằng cách thêm một chiều ở đầu tiên của tensor. Điều này thường cần thiết để đảm bảo tensor có kích thước phù hợp cho quá trình chuẩn bị dữ liệu cho mô hình.

**Bước 4: Thay đổi kích thước ảnh**

|  |
| --- |
| resized = tf.compat.v1.image.resize\_bilinear(  dims\_expander, [input\_height, input\_width]  ) |

* **resized**: Thay đổi kích thước ảnh thành kích thước mong muốn (input\_height và input\_width) bằng cách sử dụng tf.compat.v1.image.resize\_bilinear.

**Bước 5: Chuẩn hóa dữ liệu**

|  |
| --- |
| normalized = tf.divide(tf.subtract(resized, [input\_mean]), [input\_std]) |

* **normalized**: Chuẩn hóa ảnh bằng cách trừ giá trị trung bình (input\_mean) và chia cho độ chênh lệch (input\_std).

**Bước 6: Chạy phiên TensorFlow và trả kết quả**

|  |
| --- |
| sess = tf.compat.v1.Session()  return sess.run(normalized) |

* **sess**: Tạo một phiên TensorFlow (Session) để chạy phép toán trong đồ thị.
* **sess.run(normalized)**: Chạy phép toán chuẩn hóa và trả về kết quả. Kết quả này là tensor đã được chuẩn hóa và có thể được sử dụng làm dữ liệu đầu vào cho mô hình.

# 4. Định nghĩa hàm load\_labels(label\_file)

Hàm này đọc các nhãn từ một file và trả về một danh sách các nhãn.

**Bước 1: Đọc nội dung của file nhãn**

|  |
| --- |
| proto\_as\_ascii\_lines = tf.io.gfile.GFile(label\_file).readlines() |

* **tf.io.gfile.GFile(label\_file)**: Mở file nhãn bằng cách sử dụng tf.io.gfile.GFile. Thư viện TensorFlow thường được sử dụng để thao tác với tệp tin trong môi trường phân tán.
* **.readlines()**: Đọc tất cả các dòng trong file và lưu chúng vào một danh sách.

**Bước 2: Xóa khoảng trắng và dấu xuống dòng từ các dòng**

|  |
| --- |
| return [l.rstrip() for l in proto\_as\_ascii\_lines] |

* **l.rstrip()**: Xóa khoảng trắng và dấu xuống dòng từ cuối mỗi dòng trong danh sách proto\_as\_ascii\_lines.
* **[...]**: Sử dụng list comprehension để tạo một danh sách mới mà mỗi phần tử là một dòng từ file nhãn đã được xử lý.

# 5. Hàm main

## 5.1. Đặt giá trị mặc định cho các biến

|  |
| --- |
| file\_name = "tensorflow/examples/label\_image/data/grace\_hopper.jpg"  model\_file = "tensorflow/examples/label\_image/data/inception\_v3\_2016\_08\_28\_frozen.pb"  label\_file = "tensorflow/examples/label\_image/data/imagenet\_slim\_labels.txt"  input\_height = 299  input\_width = 299  input\_mean = 0  input\_std = 255  input\_layer = "input"  output\_layer = "InceptionV3/Predictions/Reshape\_1" |

## 5.2. Parse các đối số dòng lệnh bằng argparse

|  |
| --- |
|  |

## 5.3. Gán giá trị từ các đối số dòng lệnh vào biến tương ứng

|  |
| --- |
|  |

5.4.

|  |
| --- |
|  |

|  |
| --- |
|  |