Bài 29 - Xây dựng Flask API cho mô hình deep learning

23 Mar 2020 - phamdinhkhanh

Menu

1. Nhu cầu deploy model

Quá trình huấn luyện mô hình mới chỉ tạo ra các sản phẩm chạy được trên jupyter notebook. Có một AI engineer khá nổi tiếng nói rằng: `model trên jupyter notebook là model chết`. Mình khá đồng tình với quan điểm này vì nếu không đưa sản phẩm lên production thì mọi việc chúng ta làm đều `unusable`.

Với các doanh nghiệp lớn, để đưa được model vào ứng dụng sẽ cần quá trình POC, DEV, stress test, QC, deploy. Các quá trình như stress test, QC, deploy sẽ không được thực hiện bởi data scientist mà được thực hiện bởi các data engineer, QA. Chính vì thế khả năng triển khai một ứng dụng lên production luôn là một điểm hạn chế của data scientist. Thực tế kĩ năng này khá cần thiết nếu bạn muốn tự xây dựng những ứng dụng về AI.

Để triển khai được các sản phẩm về AI thì bạn sẽ cần có các kĩ năng sau ngoài kĩ năng chính về dữ liệu và mô hình.

- Kỹ năng thiết kế API back end services cho các ứng dụng.
- Kỹ năng front end.
- Kỹ năng devops để triển khai hạ tầng và môi trường cho các ứng dụng.

Các kỹ năng đó đều là những kỹ năng hoàn toàn không liên quan đến kiến thức về mô hình và thuật toán mà bạn được học. Nhưng là những kỹ năng cần thiết để làm được một sản phẩm trên production.

Chính vì vậy, nhằm giải quyết khó khăn cho data scientist. Bài viết này mình sẽ giới thiệu phương pháp xây dựng các API cho một ứng dụng AI trên python theo hướng tiếp cận đơn giản nhất.

Code cho project này được đặt tại khanhBlogTurtorial (https://github.com/phamdinhkhanh/khanhBlogTurtorial/tree/master/flask). Bạn đọc download về và thực hành theo hướng dẫn tại README.

Ngoài ra hãy join group AICode (https://www.facebook.com/groups/3235479620010379/), một group mới của mình để cùng thảo luận, tìm kiếm và chia sẻ về các source code hay.

Nội dung chính của bài này sẽ tập trung giải thích về code ở project git trên trên.

2. Project

2.1. Bài toán dự báo

Mình sẽ không train lại model từ đầu vì đã có rất nhiều các bài viết và hướng dẫn về cách huấn luyện model. Để tiết kiệm thời gian, mình sẽ sử dụng một pretrain model cho bộ dữ liệu imagenet gồm 14 triệu ảnh của 1000 classes khác nhau.

Mô hình dựa trên kiến trúc RestNet50. Một trong những kiến trúc khá hiệu quả đối với image classification. Về thuật toán RestNet (https://arxiv.org/abs/1512.03385.pdf) bạn đọc quan tâm đến kiến trúc mô hình có thể đọc thêm tại link trích dẫn bài báo gốc. Đây là thuật toán được nhóm tác giả đến từ microsoft nghiên cứu từ những năm 2015 với điểm mấu chốt là sử dụng những kết nối tắt giúp cải hiện accuracy.

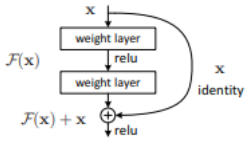


Figure 2. Residual learning: a building block.

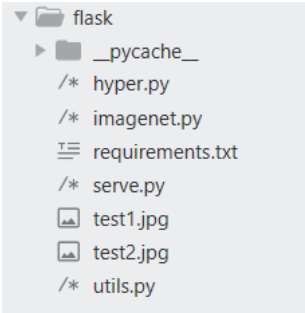
Hình 1 : Residual learning: a building block (source: từ bài báo gốc). $\mathbf{x}$ là input của mô hình. Sau khi đi qua các CNN projection, mô hình sẽ thu được ở output là $\mathcal{F}(\mathbf{x})$. Thay vì sử dụng chiến lược tăng thêm độ sâu cho model như các cách cải thiện truyền thống trước đó ở các mạng VGG. Mô hình sử dụng một kết nối xác định (mapping identity) từ $\mathbf{x}$ tới output $\mathcal{F}(\mathbf{x})$ bằng cách cộng trực tiếp vào output ở mỗi một block. Kết nối sẽ nhảy qua một vài layers.

Ngoài ra tùy vào số lượng các `building block` mà resnet có các version như RestNet30, RestNet50, RestNet100. Các mô hình này đều đã có pretrain trên imagenet. Chỉ load vào là dùng được. Tiếp theo ta sẽ cùng tìm hiểu về project.

Top

2.2. Cấu trúc project

Project của chúng ta sẽ có cấu trúc như sau:



Trong đó các file có chức năng khái quát như sau:

- `serve.py` : Khởi tạo và run app. Tại đây chúng ta khởi tạo các api cho project.
- `utils.py` : File chứa các hàm tiện ích được sử dụng ở nhiều module trong project.
- `hyper.py` : Đây là file chứa siêu tham số của project.
- `imagenet.py` : Chứa dictionary mapping index tương ứng với label của các classes trong imagenet.

Tiếp theo ta sẽ cùng nghiên cứu chi tiết của từng file này và thử vận hành project nhé.

2.3. Khởi tạo một flask app

2.3.1. Giới thiệu về Flask và API.

Flask đã quá quen thuộc đối với những ai làm web, app. Đây là một framework hỗ trợ xây dựng các API trên nền tảng python.

Flask là framework không quá nhanh. Nếu bạn làm việc nhiều với server, có nhiều ngôn ngữ thích hợp hơn mà các chuyên gia hay dùng đó là vuejs, golang, erlang. Nhưng trên ngôn ngữ python thì mình nghĩ Flask là đủ. Ngoài ra Flask cũng có ưu điểm đó là tính bảo mật của nó rất cao, code đẹp và dễ hiểu.

Về định nghĩa của API thì các bạn biết nó là gì rồi chứ? Nếu bạn chưa biết thì mình giải thích đơn giản như sau:

API được coi như chất liệu để làm nên một website. Khi chúng ta tương tác với một ứng dụng là tương tác với dữ liệu của ứng dụng đó được render trong một template html. Dữ liệu đó đến từ đâu? Nó được truyền đến thiết bị của bạn thông qua chính API. API là chiếc cầu nối dữ liệu qua lại giữa client và server và thay đổi những gì mà chúng ta nhìn thấy trên front end. Các phương thức tương tác dữ liệu trên API chính bao gồm: `GET, POST, PUT, PATCH, DELETE` . Đối với những beginner thì mình nghĩ chỉ cần hiểu `GET, POST` là đủ.

- `GET` : Client nhận dữ liệu từ server.
- `POST` : Client gửi dữ liệu lên server. Server xử lý dữ liệu và trả về một kết quả.

2.3.2. Khởi tạo API trên flask

Để khởi tạo các API trên flask chúng ta sẽ phải khởi tạo app trước. Sau đó tương ứng với mỗi API chúng ta sẽ khai báo 3 thành phần:

- route: Địa chỉ url của api.
- method: Phương thức để tương tác với api. VD: Nếu bạn muốn gửi dữ liệu lên server thì là `POST` , muốn nhận dữ liệu từ server thì là `GET` .
- hàm xử lý dữ liệu: Quyết định dữ liệu sẽ được xử lý như thế nào và trả ra cho client những gì?

Bạn có thể hình dung những gì mình nói ở trên qua ví dụ đơn giản sau:

```
1     from flask import Flask, request
2
3     # Khởi tạo flask app
4     app = Flask(__name__)
5     # route và method
6     @app.route("/", methods=["GET"])
7     # Hàm xử lý dữ liệu
8     def _hello_world():
9             return "Hello world"
```

Chỉ cần như thế là bạn đã khởi tạo được một app trên flask rồi đó. Tất nhiên đối với những dự án lớn thì sẽ cần nhiều thứ phức tạp hơn như bảo mật, design pattern api, blabla,.... Mình sẽ không đi sau vì không phải là dân chuyên về lĩnh vực này.

2.4. Tìm hiểu API trong project thực hành

Trong code của project bạn mở file `serve.py` có nội dung như bên dưới:

Top

```
from PIL import Image
import numpy as np
import hyper as hp
from flask import Flask, request
import flask
import json
import io
import utils
import imagenet

# Khởi tạo model.
global model
model = None
# Khởi tạo flask app
app = Flask(__name__)

# Khai báo các route 1 cho API
@app.route("/", methods=["GET"])
# Khai báo hàm xử lý dữ liệu.
def _hello_world():
        return "Hello world"

# Khai báo các route 2 cho API
@app.route("/predict", methods=["POST"])
# Khai báo hàm xử lý dữ liệu.
def _predict():
        data = {"success": False}
        if request.files.get("image"):
                # Lấy file ảnh người dùng upload lên
                image = request.files["image"].read()
                # Convert sang dạng array image
                image = Image.open(io.BytesIO(image))
                # resize ảnh
                image_rz = utils._preprocess_image(image,
                        (hp.IMAGE_WIDTH, hp.IMAGE_HEIGHT))
                # Dự báo phân phối xác suất
                dist_probs = model.predict(image_rz)
                # argmax 5
                argmax_k = np.argsort(dist_probs[0])[::-1][:5]
                # classes 5
                classes = [imagenet.classes[idx] for idx in list(argmax_k)]
                # probability of classes
                classes_prob = [dist_probs[0, idx] for idx in list(argmax_k)]
                data["probability"] = dict(zip(classes, classes_prob))
                data["success"] = True
        return json.dumps(data, ensure_ascii=False, cls=utils.NumpyEncoder)

if __name__ == "__main__":
        print("App run!")
        # Load model
        model = utils._load_model()
        app.run(debug=False, host=hp.IP, threaded=False)
```

Nhìn file `serve.py` bạn thấy gì?

Bỏ qua các tiểu tiết xử lý bên trong. Nhìn khái quát các hàm thì ta có thể thấy đây chính là file thiết kế các API của project này. Đồng thời ta cũng nhận định đây là file execute chính của project vì nó có điều kiện `__main__` ở cuối.

Cụ thể hơn trong file này các bạn có thể thấy chúng ta thực hiện các chức năng sau:

- Khởi tạo Flask application: dòng `app = Flask(__name__)`.
- Khai báo các đường link của api thông qua các câu lệnh `@app.route()`. Nếu các bạn hiểu sâu hơn thì đây chính là `decorator` trong python có tác dụng bổ sung thêm chức năng cho hàm bên dưới.
- Các hàm xử lý dữ liệu của API như `_predict()`, `_hello_world()`.

Chúng ta cần chú ý tới hàm số `_predict()`. Bên trong hàm này ta sẽ nhận file dữ liệu ảnh được gửi lên từ client, sau đó parse sang định dạng numpy array. Sử dụng model được khai báo global để dự báo kết quả. Chi tiết chức năng của từng lệnh mình đã comment trong file.

Dành cho bạn nào là begginer: Khi chúng ta chạy file bằng câu lệnh:

```
python server.py
```

Chương trình sẽ thực thi các lệnh trong điều kiện `if __name__ == "__main__":` Đầu tiên. Do đó model sẽ được load đầu tiên. Do ở đây mình không dùng lệnh `async await` nên chương trình sẽ chờ đến khi load model xong thì mới chạy lệnh bên dưới. Python sẽ hơi khác so với các ngôn ngữ compile như java hay C, C++ một chút bởi vì nó mặc định là xử lý đơn luồng nên sẽ chạy tuần tự từ trên xuống dưới. Ngoài python thì còn một ngôn ngữ rất nổi tiếng khác cũng chạy đơn luồng, đó là javascript.

Sau khi load xong model thì chương trình sẽ start app thông qua lệnh `app.run()`. Lúc này bạn đã có thể tương tác được với các api của nó.

2.5. Xử lý code theo design pattern

Design pattern là một kiến thức rất quan trọng của lập trình hướng đối tượng. Hiểu đơn giản thì design pattern là các mẫu thiết kế có sẵn để mình code project sao cho nó đẹp mắt, gọn gàng và thuận tiện.

Top

Ở đây mình sẽ không đi sâu vào các kiểu design pattern trong lập trình OOP vì các bạn có thể google ra hàng ngàn tài liệu về domain quan trọng này. Mình chỉ nêu ra một số kinh nghiệm mà mình áp dụng ở đây:

- Luôn để các hàm chức năng ở một file gọi là `utils.py` . Trong project này file `utils.py` như sau:

```
import hyper as hp
from tensorflow.keras.applications import MobileNet, ResNet50
from tensorflow.keras.preprocessing.image import img_to_array
import numpy as np
import json

# Load model
def _load_model():
        # Khởi tạo model
        model = ResNet50(weights='imagenet')
        print("Load model complete!")
        return model


# Resize ảnh
def _preprocess_image(img, shape):
        img_rz = img.resize(shape)
        img_rz = img_to_array(img_rz)
        img_rz = np.expand_dims(img_rz, axis=0)
        return img_rz

# Encoding numpy to json
class NumpyEncoder(json.JSONEncoder):
    '''
    Encoding numpy into json
    '''
    def default(self, obj):
        if isinstance(obj, np.ndarray):
            return obj.tolist()
        if isinstance(obj, np.int32):
            return int(obj)
        if isinstance(obj, np.int64):
            return int(obj)
        if isinstance(obj, np.float32):
            return float(obj)
        if isinstance(obj, np.float64):
            return float(obj)
        return json.JSONEncoder.default(self, obj)
```

Ta thấy trong file `utils` sẽ để phần lớn là các hàm thường xuyên được dùng trong toàn project như load, save model dữ liệu, preprocessing data và các hàm encoding, decoding,.... Tóm lại những hàm thông dụng, thường xuyên được sử dụng ở nhiều files khác nhau thì nên cấu trúc nó vào `utils` . Nếu bạn thắc mắc utils là gì? Nó chính là viết tắt của cụm từ `utilities` tức là các hàm `tiện ích` .

- Luôn lưu các parameters của project vào một file là `hyper.py` (viết tắt của hyperparameter - siêu tham số). Những project to thì sẽ xuất hiện càng nhiều các tham số. Việc đặt các siêu tham số ở chung một file có lợi ích là nó xuất hiện ở n files khác nhau, bạn không phải tìm mất công mà chỉ phải về file hyper tìm và sửa nó. Việc không đặt parameters cho các hằng số như input shape, số units của các layers,... sẽ dễ khiến các bạn kiểm soát dự án của mình khó khăn hơn.
- Luôn để các packages mà mình sử dụng trong project vào file `requirements.txt` . Khi chúng ta code một project trên local thì môi trường là các packages mà ta sử dụng ở máy tính của mình. Nhưng khi chuyển giao code cho người khác chưa chắc máy của họ đã có những packages đó. Do đó việc đưa các package name vào `requiments.txt` giúp người khác cài đặt môi trường nhanh hơn. Đã support là phải support nhiệt tình, đã làm là phải làm đến nơi đến chốn.
- Nên cài đặt một virtual environment cho mỗi một project: Mục đích là để tránh các lỗi xung đột về version. Khi bạn tải code của người khác về và triển khai trên virtual enviroment những package mà có version khác với cấu hình mặc định của python của bạn thì những package đó chỉ sống trong virtual enviroment thôi. Bạn sẽ không cần phải uninstall lại version mặc định của chúng trên python và không ảnh hưởng đến các project khác.

Và còn nhiều những kinh nghiệm khác mà các bạn khi làm việc với càng nhiều project thì sẽ tự đúc rút ra kinh nghiệm.

3. Vận hành app

Như vậy là mình đã giới thiệu qua các file `serve.py` , `utils.py` và `hyper.py` với chức năng của từng file. Ngoài ra còn một file nữa là `imagenet.py` . File này chỉ có tác dụng lưu trữ label của các classes trong bộ dữ liệu imagenet theo index tương ứng.

Cuối cùng để tận hưởng thành quả đọc bài miệt mài từ đầu đến cuối. Bạn chạy lện sau trên commanline:

```
python serve.py
```

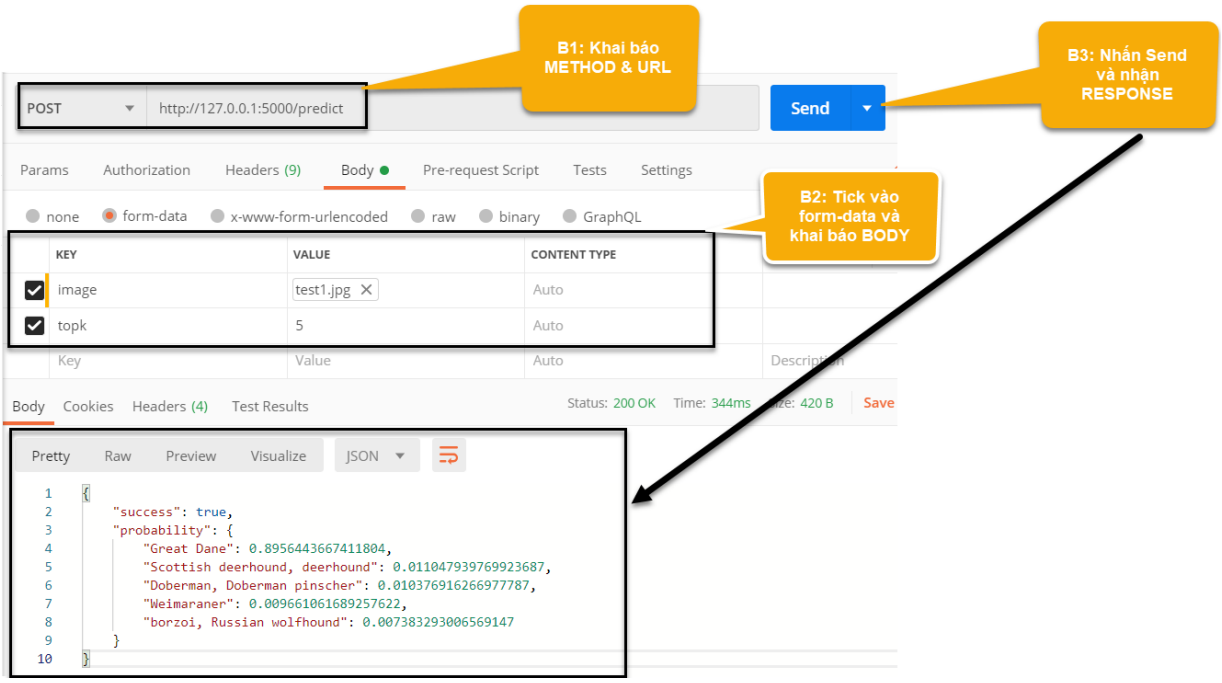
Khi thành công sẽ xuất hiện dòng `Running on http://127.0.0.1:5000/` . Chúng ta có thể kiểm tra chức năng của API bằng lện curl (nếu bạn dùng ubuntu, macintosh, linux).

Muốn kiểm tra chức năng của API thì có thể dùng lệnh curl.

```
curl -X POST -F image=@test1.jpg 'http://127.0.0.1:5000/predict'
```

Với các bạn sử dụng window thì download postman (https://www.postman.com/downloads/) và làm theo hướng dẫn như ảnh bên dưới:

Top



Ở bước 2 nhớ là phải import hẳn file ảnh vào thì mới được.

Như vậy là các bạn đã hoàn thành một project nho nhỏ về thiết kế một API dự báo nhãn cho hình ảnh. Project này không có gì phức tạp về mặt techinical, chỉ hướng tới việc hiểu và thực hành để build được các API trên flask. Xa hơn chúng ta có thể áp dụng các kiến thức này để làm những ứng dụng hay ho về AI.

Bài viết của mình xin tạm dừng tại đây. Các bạn có thể download code của nó tại project KhanhBlogTurtorial (https://github.com/phamdinhkhanh/khanhBlogTurtorial/tree/master/flask).

Nếu bạn gặp lỗi cần hỗ trợ hãy post lên group AICode (https://www.facebook.com/groups/3235479620010379/) của mình để cùng trao đổi.

3. Tài liệu

1. Flask - quick start (https://flask.palletsprojects.com/en/1.1.x/quickstart/)
2. Flask REST API - Youtube (https://www.youtube.com/watch?v=s_ht4AKnWZg)
3. Design patterns elements reusable object oriented (https://www.amazon.co.uk/Design-patterns-elements-reusable-object-oriented/dp/0201633612)
4. Head first design patterns - blog dịch (https://toihocdesignpattern.com/mo-dau-head-first-design-patterns-tieng-viet.html)
5. Design pattern for dummies (https://www.academia.edu/8305787/Series_b%C3%A0i_d%E1%BB%8Bch_Design_Pattern_for_Dummies_Series_b%C3%A0i_d%E1%BB%8Bch_Desig

Top