

# TRỰC QUAN HÓA MẠNG CNN

Nguyễn Vinh Tiệp và các cộng sự VietAI

Ngày 9 tháng 9 năm 2018

## Tóm tắt nội dung

Bài tutorial này tiến hành trực quan hóa các kết quả của một mạng CNN, cụ thể là mạng VGG16. Để làm điều này, chúng tôi tiến hành thí nghiệm trên mạng VGG16 đã được huấn luyện trước đó trên tập dữ liệu ImageNet [1]. Khi đưa vào một số ảnh với các tính chất khác nhau, ta sẽ quan sát và giải thích các kết quả trả về. Kết quả thực nghiệm cho thấy mạng CNN có thể biểu diễn được các đặc trưng có yếu tố ngữ nghĩa cao.

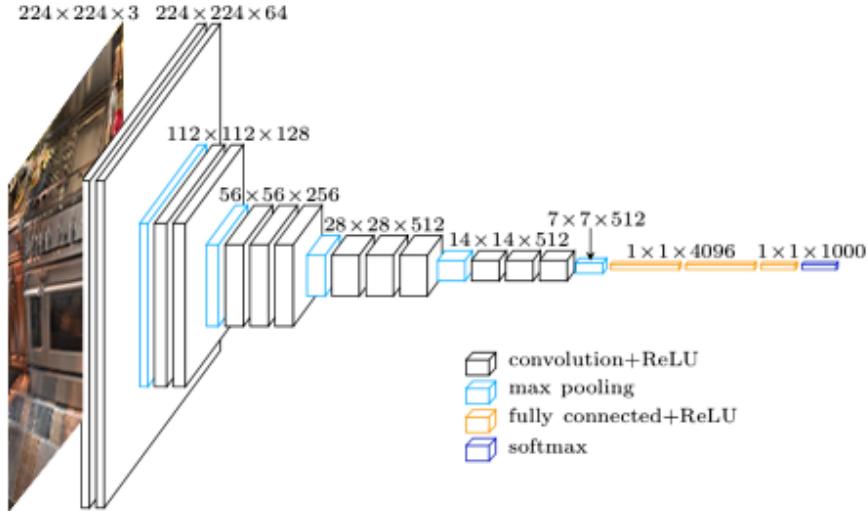
## 1 Giới thiệu kiến trúc mạng VGG16

Mạng VGG16 được đề xuất bởi Simonyan vào năm 2015 trong hội nghị ICLR [2]. Đây là một kiến trúc mạng tương đối gọn nhưng vẫn cho kết quả tốt và ổn định so với các kiến trúc mạng trước đó. Đây chính là nguyên nhân chính để chúng tôi sử dụng mạng này phục vụ cho việc trực quan hóa.

Như tên gọi đã thể hiện, mạng này bao gồm 16 lớp chính với các phép biến đổi tích chập (từ đây chúng tôi xin gọi là Convolution) và kết nối đầy đủ (từ đây chúng tôi xin gọi là Fully Connect hay FC). Chú ý rằng, mặc định thì sau khi thực hiện phép biến đổi Convolution ta sẽ phải đi qua hàm kích hoạt (activation function) ngay sau đó. Riêng phép biến đổi *max pooling* không được tính là phép biến đổi chính theo quan điểm của các tác giả. Hình 1 minh họa kiến trúc của mạng VGG16. Mạng này được chia thành các nhóm sau:

- Nhóm Conv1: bao gồm 2 phép biến đổi Convolution liên tiếp và một phép max pooling.
- Nhóm Conv2: bao gồm 2 phép biến đổi Convolution liên tiếp và một phép max pooling.
- Nhóm Conv3: bao gồm 3 phép biến đổi Convolution liên tiếp và một phép max pooling.
- Nhóm Conv4: bao gồm 3 phép biến đổi Convolution liên tiếp và một phép max pooling.
- Nhóm Conv5: bao gồm 3 phép biến đổi Convolution liên tiếp và một phép max pooling.

- Nhóm FC6, FC7 và FC8: bao gồm 3 phép kết nối đầy đủ liên tiếp với 4096 neuron cho FC6, FC7 và 1000 neuron đầu ra cho lớp FC8.



Hình 1: Kiến trúc mạng VGG16.

Bài tutorial này sử dụng các tham số của mô hình này sau khi được huấn luyện với tập dữ liệu ImageNet bao gồm 1.000 lớp đối tượng. Chúng tôi sử dụng và kế thừa mã nguồn từ trang Github của tài khoản *machrisaa*<sup>1</sup>. Tuy nhiên để đơn giản, chúng tôi không sử dụng mạng VGG19 mà chỉ sử dụng mạng VGG16 để trực quan hóa. Việc cài đặt thuật toán truyền thuận cho mạng VGG16 như mô tả ở trên được thực hiện trong phương thức **build(self, rgb)** của file **vgg16.py**

## 2 Sử dụng mạng VGG đã huấn luyện để phân loại đối tượng

**vis\_vgg16\_sec2.py:** Đầu tiên, để cảm nhận kết quả trả về khi đưa ảnh của một đối tượng vào mạng CNN, chúng ta sử dụng ảnh của một con hổ được đặt ở đường dẫn "test\_data/tiger.jpeg" (Hình 2). Ta sẽ lần lượt thực hiện các bước sau:

**Bước 1:** Sử dụng hàm **utils.load\_image** để đọc và chuyển đổi (resize) về kích thước  $224 \times 224 \times 3$ . Sau đó ta tiến hành chuyển sang dạng khối (batch) để có thể đưa vào mạng VGG nhiều ảnh cùng một lúc. Tuy nhiên, với ví dụ đầu tiên chúng tôi chỉ sử dụng duy nhất một ảnh nên kích thước batch lúc này

<sup>1</sup><https://github.com/machrisaa/tensorflow-vgg>

bằng 1. Do đó dữ liệu đưa vào mạng VGG16 sẽ là một tensor có kích thước  $1 \times 224 \times 224 \times 3$ . Biến 'batch' được sử dụng để lưu ảnh đầu vào.



Hình 2: Ảnh đầu vào với một đối tượng duy nhất - Con hổ.

```
1 img = utils.load_image("./test_data/tiger.jpeg")
2 batch = img.reshape((1, 224, 224, 3))
```

**Bước 2:** Khởi tạo đối tượng *placeholder* của thư viện Tensorflow để định nghĩa dữ liệu đầu vào cho mạng. Lưu ý rằng, kích thước của placeholder này phải đồng nhất với kích thước của 'batch'. Đồng thời chúng ta tạo sẵn một *feed\_dict* để sau này truyền dữ liệu "thật" vào placeholder.

```
1 images = tf.placeholder("float", [1, 224, 224, 3])
2 feed_dict = {images: batch}
```

**Bước 3:** Load mạng đã huấn luyện trước đó vào đối tượng thuộc lớp *vgg16*:

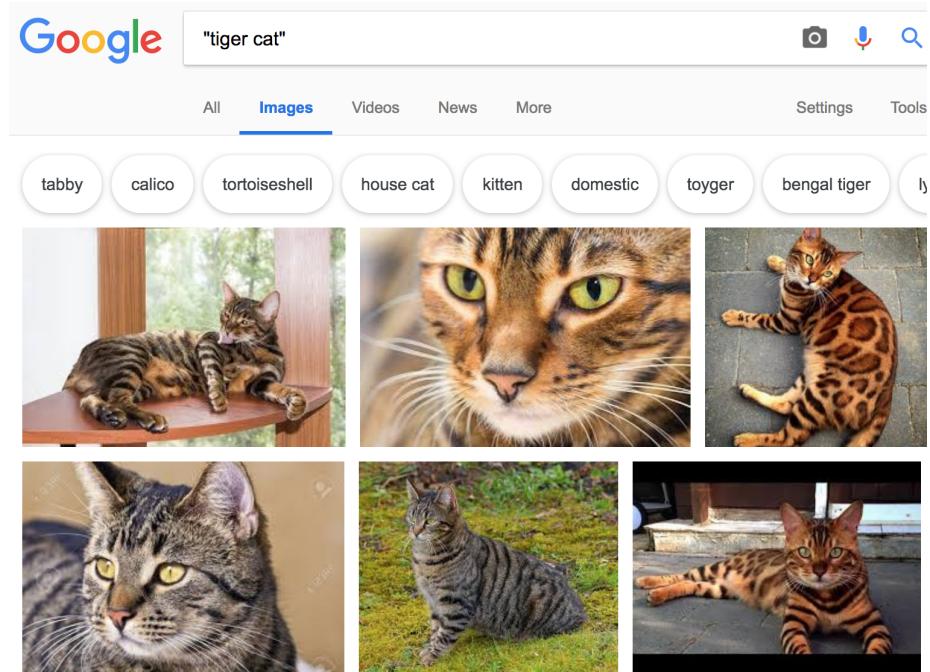
```
1 vgg = vgg16.Vgg16()
2 with tf.name_scope("content_vgg"):
3     vgg.build(images)
```

**Bước 4:** Tiến hành đưa dữ liệu vào mạng VGG16 và trả về kết quả tính toán sau khi thực hiện lớp soft-max cuối cùng. Kết quả trả về biến 'prob' chính là giá trị xác suất theo từng lớp đối tượng được quy ước trước đó. Danh sách các lớp đối tượng của tập dữ liệu ImageNet được lưu trong file 'synset.txt'. Chúng ta sử dụng hàm *utils.print\_prob* để in ra 5 lớp đối tượng "gần giống" với ảnh đầu vào nhất.

```
1 prob = sess.run(vgg.prob, feed_dict=feed_dict)
2 print("Top 5 object gan giong voi hinh tiger.jpeg:")
3 utils.print_prob(prob[0], './synset.txt')
```

Kết quả in ra màn hình console như hình bên dưới. Kết quả cho thấy, loại đối tượng gần giống nhất được gán nhãn là 'n02129604 tiger, Panthera tigris'

với giá trị xác suất phân loại là 0.82691574. Trong đó, 'n02129604' là mã số loại đối tượng, 'tiger' (hổ) là tên viết tắt và 'Panthera tigris' là tên khoa học của đối tượng. Loại đối tượng gần giống thứ hai chính là 'tiger cat' (mèo lông vằn - giống con hổ) với xác suất phân loại là 0.1713691. Sau đó ta thử Google với từ khoá "tiger cat" thì cho ra kết quả là những ảnh có **hình dạng giống với ảnh đầu vào**. (Hình 3). Như vậy ta có thể thấy là vector xác suất trả về là hợp lý.



Hình 3: Kết quả tìm kiếm với từ khoá "tiger cat".

```

1 Top 5 object gan giong voi hinhanh tiger.jpeg:
2 ('Top5: ', [( 'n02129604 tiger', 'Panthera tigris', 0.82691574), ( 'n02123159 tiger cat', 0.1713691), ( 'n02128925 jaguar', 'panther', 'Panthera onca', 'Felis onca', 0.0012765457), ( 'n02127052 lynx', 'catamount', 0.00017351758), ( 'n02128385 leopard', 'Panthera pardus', 0.00015544154)])

```

### 3 Thủ nghiệm trên hai đối tượng

**vis\_vgg16\_sec3.py:** Hoàn toàn tương tự như thí nghiệm ở mục trước, trong mục này chúng tôi sẽ thử sử dụng nhiều ảnh đầu vào với hai đối tượng chính cùng một lúc. Hình đầu tiên, chúng tôi sử dụng ảnh có hai đối tượng có **kích thước ngang nhau** là 'test\_data/dog-and-cat.jpg' bao gồm chó và mèo như Hình 4. Hình thứ hai, chúng tôi sử dụng ảnh hai đối tượng có kích thước chênh

lệch nhau đáng kể là 'test\_data/elephant-and-dog.jpg' bao gồm voi và chó như Hình 5.



Hình 4: Ảnh đầu vào gồm 2 đối tượng có kích thước ngang nhau: chó và mèo.



Hình 5: Ảnh đầu vào gồm 2 đối tượng có kích thước chênh lệch: voi và chó.

Để tiến hành đưa vào mạng CNN nhiều ảnh cùng một lúc, ở Bước 1, chúng ta cần sửa lại mã nguồn như sau.

```
1 img1 = utils.load_image("./test_data/dog-and-cat.jpg")
2 img2 = utils.load_image("./test_data/elephant-and-dog.jpg")
3
4 batch1 = img1.reshape((1, 224, 224, 3))
5 batch2 = img2.reshape((1, 224, 224, 3))
6
```

```
7 batch = np.concatenate((batch1, batch2), 0)
```

Kết quả trả về vẫn là một biến 'prob' bao gồm 2 mảng chứa giá trị xác suất phân loại cho từng ảnh khi đưa vào thí nghiệm. Để in ra top 5 đối tượng gần giống nhất, ta viết mã nguồn như sau:

```
1 prob = sess.run(vgg.prob, feed_dict=feed_dict)
2 print("Top 5 object gan giong voi hinhanh dog-and-cat:")
3 utils.print_prob(prob[0], './synset.txt')
4 print("Top 5 object gan giong voi hinhanh elephant-and-dog:")
5 utils.print_prob(prob[1], './synset.txt')
```

Kết quả phân loại đối tượng với ảnh đầu tiên (chó và mèo) được thể hiện như ở bên dưới đây. Ta thấy rằng trong 5 top đối tượng đầu tiên thì có đến 4 đối tượng thuộc giống chó (beagle, Brittany spaniel, bluetick và Walker hound) và 1 đối tượng thuộc giống mèo (Egyptian cat). Có thể giải thích rằng, phần diện tích ảnh chứa đối tượng chó nhỉnh hơn mèo một chút và đặc trưng của chó và mèo cũng tương đối giống nhau.

```
1 Top 5 object gan giong voi hinhanh tiger.jpeg:
2 ('Top5:', [('n02088364 beagle', 0.36113298), ('n02101388 Brittany
   spaniel', 0.10962392), ('n02088632 bluetick', 0.0923439), ('n02124075 Egyptian cat', 0.030632038), ('n02089867 Walker hound
   , Walker foxhound', 0.02731003)])
```

Kết quả phân loại đối tượng với ảnh thứ hai (voi và chó) được thể hiện như ở bên dưới đây. Ta thấy rằng cả 5 đối tượng đầu tiên đều là giống voi (ví dụ như tusker và kết quả tìm kiếm với từ khóa "tusker" ở Hình 6) hoặc một con vật bị nhầm lẫn với voi là bò rừng (bison) (kết quả tìm kiếm với từ khóa "bison" ở Hình 7). Không có đối tượng nào liên quan đến chó được trả về trong top 5 hoặc thậm chí là top 10. Như vậy ta có thể thấy là, do kích thước của con chó quá nhỏ so với tấm ảnh nên mạng VGG đã **không nhận ra được**.

```
1 Top 5 object gan giong voi hinhanh elephant-and-dog:
2 ('Top5:', [('n01871265 tusker', 0.5439142), ('n02504013 Indian
   elephant, Elephas maximus', 0.29138574), ('n02504458 African
   elephant, Loxodonta africana', 0.15230341), ('n02410509 bison',
   0.002346075), ('n02108422 bull mastiff', 0.002085954)])
```

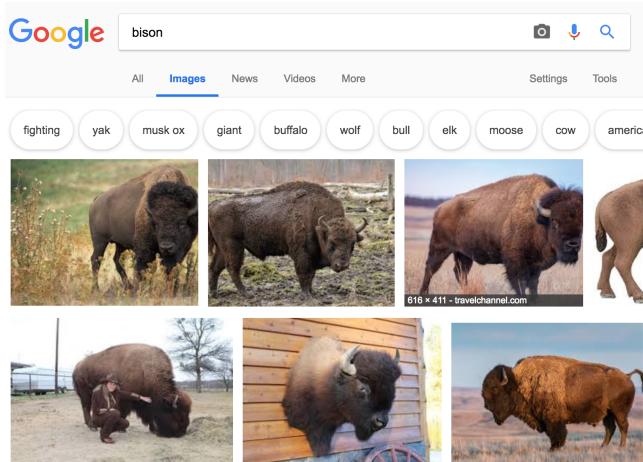
## 4 Thủ nghiệm trên đối tượng bị che khuất

**vis\_vgg16\_sec4.py:** Tiếp nối thí nghiệm trên, chúng ta sẽ thử đánh giá vai trò của từng vùng trong tấm ảnh đầu vào lên giá trị xác suất phân loại đối tượng đầu ra. Cụ thể là ta sẽ tiến hành che khuất một số vùng từ không quan trọng đến vùng quan trọng. Hoàn toàn tương tự, trong thí nghiệm này chúng ta sẽ thử trên ảnh con hổ đầy đủ (Hình ??), ảnh bị cắt một vùng không thuộc con hổ (Hình 8) và ảnh bị cắt một vùng quan trọng trên mặt con hổ (Hình 9).

Kết quả cho thấy, xác suất nhận biết đối tượng trong ảnh là con hổ lần lượt là 0.8269, 0.8169 và 0.7575. Đồng thời, xác suất nhận biết mèo lông hổ (tiger cat) cùng tăng lên từ 0.1713 lên 0.2376. Như vậy có thể thấy là, việc che đi các đặc trưng quan trọng của ảnh đầu vào sẽ khiến cho mạng CNN dễ bị nhận nhầm hơn so với lớp đối tượng khác.



Hình 6: Kết quả tìm kiếm với từ khóa "tusker".



Hình 7: Kết quả tìm kiếm với từ khoá "bison".

## 5 Trực quan hoá feature map

**vis\_vgg16\_sec5.py:** Cuối cùng, chúng tôi tiến hành trực quan hoá một số kết quả trung gian trong quá trình tính toán feature map của mạng VGG. Cụ thể là ở đây chúng tôi sẽ trích suất ra lớp conv1\_1 từ đối tượng vgg16 và hiển thị bằng thư viện OpenCV (cv2). Mã nguồn để trích xuất và hiển thị tương đối đơn giản như sau:

```
1 prob, conv1_1 = sess.run([vgg.prob, vgg.conv1_1], feed_dict=
```



Hình 8: Ảnh con hổ bị cắt phần không quan trọng.



Hình 9: Ảnh con hổ bị cắt phần quan trọng.

```
2
3 conv1_1 = conv1_1[0,:,:,:]
4 cv2.imshow('The first feature map in conv1_1',conv1_1)
5 cv2.waitKey(0)
6 cv2.destroyAllWindows()
```

Hình 10 minh họa ảnh một feature map trên mạng đã được huấn luyện trước. Khởi đầu, các trọng số của mạng VGG16 được khởi tạo ngẫu nhiên. Sau khi huấn luyện, các filter của mạng có xu hướng sẽ được dùng để lọc ra các đặc trưng biên cạnh nổi bật của tấm hình nhằm phục vụ cho bài toán phân loại đối tượng. Quá trình này được diễn ra hoàn toàn tự động và không có sự thiết kế bởi chuyên gia. Các trọng số của filter được cập nhật trong quá trình huấn luyện với khuynh hướng làm giảm độ lỗi của kết quả nhận dạng nhưng kết quả cuối cùng cho thấy sự hợp lý của mạng CNN khi đã lọc lại các thông tin về biên cạnh của đối tượng. Đây chính là các đặc trưng chủ đạo giúp phân biệt các đối tượng với nhau.



Hình 10: Trực quan hoá một feature map của lớp conv1\_1 sử dụng thư viện OpenCV.

## Tài liệu

- [1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.
- [2] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.