Bài 41 - DeepLab Sentiment Segmentation

18 Jun 2020 - phamdinhkhanh

Menu

1. Giới thiệu chung

Ở Bài 40 - Image Segmentation (https://phamdinhkhanh.github.io/2020/06/10/ImageSegmention.html) chúng ta đã được tìm hiểu về một số thuật toán Image Segmentation. Thông qua đó, chúng ta đã nắm được cơ bản về input/output, kiến trúc của một mô hình Image Segmentation và một số lớp mô hình Image Segmentation phổ biến như Mask-CNN, U-Net, FCN.

Ở bài này chúng ta sẽ tiếp tục tìm hiểu những kiến trúc Image Segmentation hiện đại hơn và hiện tại đang là những kiến trúc SOTA nhất. Hãy cùng khám phá DeepLab V1+V2 và DeepLab V3, đặc điểm kiến trúc mạng và những kỹ thuật được áp dụng bên trong nó nhé.

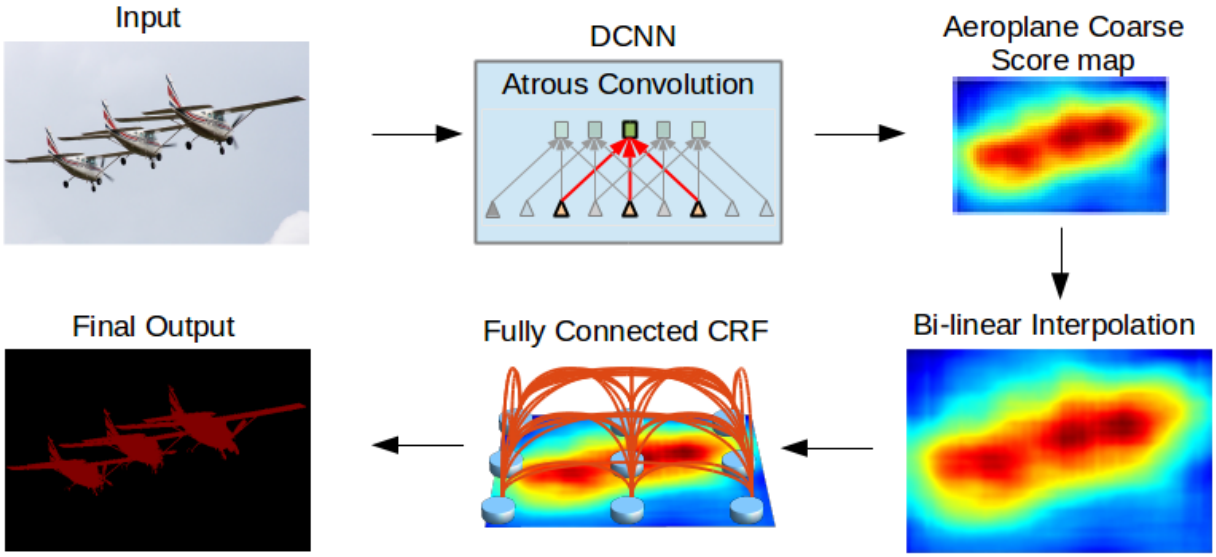
2. DeepLabV1+V2

2.1. Kiến trúc chung của DeepLab model

DeepLabV1 và V2 được nhóm tác giả `Liang-Chieh Chen, George Papandreou` là những nhà nghiên cứu lâu năm về Semantic Segmentation công bố lần đầu vào năm 2016 và lần thứ 2 vào năm 2017. Bản thảo lần 2 là bổ sung của lần 1 nên DeepLabV1 và V2 không có gì khác biệt nhiều. Ngay sau khi công bố, kiến trúc đã đạt được những kết quả ấn tượng trên tập dữ liệu thẩm định của Pascal VOC 2012. Đây là một kiến trúc áp dụng một cách rất linh hoạt tích chập Atrous thay vì các phương pháp trước đó là áp dụng Transposed Convolution. Bên cạnh đó tác giả cũng áp dụng phương pháp Conditional Random Field để tinh chỉnh kết quả dự báo chuẩn xác hơn.

Top

2.2. Kiến trúc chung của DeepLabV1+V2 model



Hình 1: Các thành phần của DeepLab model lần lượt theo hướng mũi tên từ trên xuống dưới và từ trái qua phải bao gồm: `Input -> DCNN -> Score Map -> bilinear interpolation -> Fully Connected CRF -> mask output` . Source: DeepLabV2 (https://arxiv.org/pdf/1606.00915.pdf)

Chúng ta sẽ cùng giải thích qua công dụng của từng phần trong mạng DeepLab:

- Đầu tiên một ảnh đầu vào sẽ được truyền vào một mạng CNN học sâu nhiều tầng (DCNN - Deep Convolutional Neural Network). Nhiệm vụ chính của DCNN là tạo ra một feature map là một biểu diễn không gian các đặc trưng của ảnh đầu vào. Trong DCNN chúng ta sử dụng các tích chập Atrous để trích lọc đặc trưng thay vì các tích chập CNN thông thường. Tích chập Atrous sẽ có những tác dụng đặc biệt hơn so với tích chập CNN đó là tầm nhìn (`field of view`) được mở rộng hơn, không làm giảm chiều của feature map quá sâu mà vẫn giữ được số lượng tham số và chi phí tính toán tương đương với tích chập CNN. Lý do không nên giảm kích thước feature map ở các bài toàn Image Segmentation là bởi vì việc giảm kích thước feature map có thể dẫn tới mất mát các thông tin về không gian. Trong khi mục tiêu của lớp bài toán Image Segmentation là đồng thời **định vị** vị trí pixels và dự báo nhãn cho chúng. Sau cùng của mạng DCNN ta thu được một feature map là một bản đồ đặc trưng của ảnh đầu vào chính là `Aeroplane Coarse Score map` trong hình vẽ.
- Score map có kích thước nhỏ hơn nhiều so với ảnh gốc. Chúng ta sử dụng Bi-linear Interpolation để resize lại score map về kích thước gốc. Bố cục của ảnh sau khi resize không khác so với ảnh gốc, chỉ thay đổi về kích thước.
- Để tạo ra được feature map dự báo thì chúng ta áp dụng một layer kết nối toàn bộ (Fully Connected Layer) kết hợp với phương pháp Conditional Random Field, một phương pháp thuộc nhóm mô hình đồ thị xác suất (Probabilistic Graphical Model) để chuẩn hóa lại **nhãn** cho các pixels. Sau chuẩn hóa, từ score map chúng ta thu được final output có **đường biên** và **vùng ảnh** trở nhên rõ ràng hơn.

Chúng ta có thể thấy tiến trình hoạt động của DeepLab version 1 và 2 rất đơn giản phải không nào.

Như điểm mấu chốt ở các kiến trúc này đó là áp dụng `Atrous Convolution` và `Fully Connected CRF` để dự đoán ma trận mask cho ảnh chuẩn xác hơn.

Top

Để hình dung sâu hơn về những cơ chế này chúng ta sẽ tìm hiểu phần tiếp theo.

2.3. Atrous Convolution

Atrous có nghĩa là `à trous` là một từ tiếng Pháp tương ứng với từ `hole` trong tiếng anh ám chỉ rằng có một khoảng trống giữa các tích chập. Tích chập này thường được sử dụng trong `tín hiệu sóng` . Nội dung của nó cũng tương tự như `Dilation Convolution` nhưng trong các thuật toán Image Segmentation hiện đại thì đa phân sử dụng `Atrous Convolution` thay cho `Dilation Convolution` , trên thực tế thì hai khái niệm này tương đương. Ở bài trước mình đã trình bày dilation convolution (https://phamdinhkhanh.github.io/2020/06/10/ImageSegmention.html#8-t%C3%ADch-ch%E1%BA%ADp-gi%C3%A3n-n%E1%BB%9F-dilation-convolution) trong trường hợp thêm xen kẽ dòng, cột 0 với kích thước bằng 1.

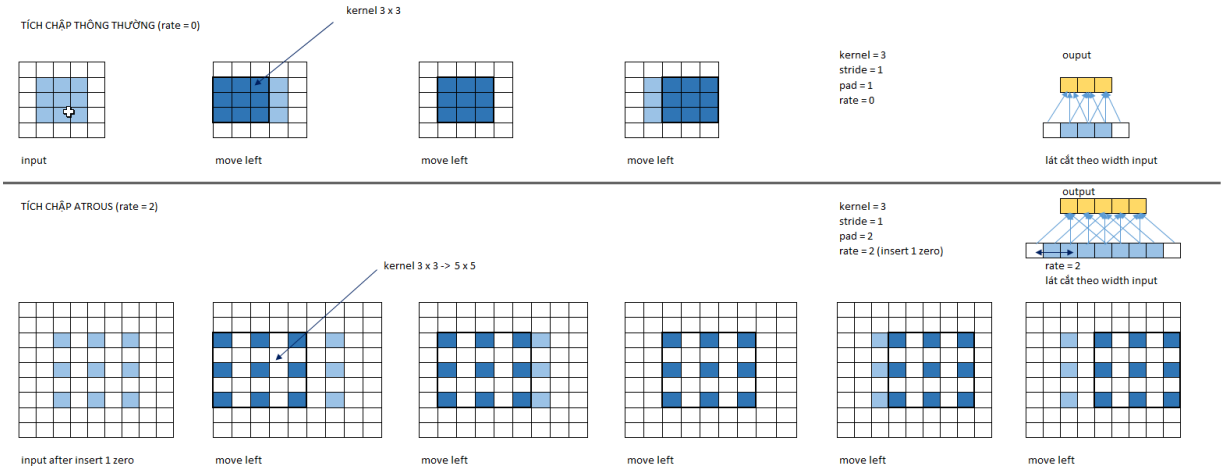
Đây là một tích chập thường được sử dụng trong các bài toán image segmentation. Tích chập atrous cho phép chúng ta trích lọc được các đặc trưng ở mật đồ dày hơn khi thông tin được bảo toàn tốt hơn cho các đối tượng ở những kích thước khác nhau.

Chúng ta có thể khái quát Atrous Convolution trong trường hợp tổng quát thông qua công thức:

$$y[i] = \sum_{i=1}^{K} x[i + r.k]w[k]$$

Ý nghĩa của công thức trên là: Với mỗi một cell i trên output y, tích chập atrous sẽ tính toán bằng cách nhân tích chập bộ lọc w với feature map x. Ở đây atrous rate r tương ứng với khoảng cách mà ta giãn cách giữa các dòng và cột bằng các giá trị 0, nếu atrous rate bằng r thì giãn cách các dòng và cột liên tiếp là $r - 1$ dòng, cột.

Để hình dung rõ hơn các bạn theo dõi hình minh họa bên dưới.



Hình 2: Tích chập thông thường (bên trên) và tích chập atrous (bên dưới) với cùng một bức ảnh 3 x 3. Các ô màu xám là ảnh gốc, màu trắng là padding và dòng, cột 0 được thêm vào và màu xanh nước biển là các vùng nhận thức (receptive field) khi nhân tích chập. Bên phải ngoài cùng là mô tả phép chiếu của lát cắt theo width quá trình thực hiện tích chập ở mỗi bước. Ta thấy đối với tích chập atrous thì các vị trí của receptive field giãn cách nhau một cell mặc dù về bản chất vẫn là tích chập với bộ lọc `3 x 3` nhưng được thực hiện trên một vùng rộng hơn là `5 x 5` . Tích chập thông thường thì kích thước receptive field bằng với kích thước bộ lọc và bằng `3 x 3` . Kết quả output cho thấy tích chập astrous đã tăng kích thước feature map từ `3x3` lên `5x5` .

Để thực hiện tích chập atrous như hình minh họa đầu tiên chúng ta giãn cách các pixels trên ảnh gốc bằng cách thêm các dòng và cột 0 xen kẽ nhau. Sau đó ta thêm padding bằng 2 về các phía trái, phải, trên và dưới. Thực hiện di chuyển tích chập 1 cell theo chiều từ trái qua phải và từ trên

Top

xuống dưới trên các vùng receptive field kích thước `3 x 3` nhưng đã được giãn cách thành `5 x 5` . Cuối cùng ta thu được output có kích thước là `5 x 5` .

Như vậy tích chập astrous có tác dụng:

- Tăng kích thước output so với input.
- Vùng nhận thức lớn hơn giúp mở rộng tầm nhìn của bộ lọc và phù hợp với các bối cảnh rộng hơn.

Đây là một cơ chế cho phép kiểm soát tầm nhìn của vùng nhận thức và tìm ra sự đánh đổi hợp lý nhất giữa độ chính xác cục bộ (áp dụng với tầm nhìn hẹp) và sự đồng nhất bối cảnh (áp dụng với tầm nhìn rộng).

Xin trích dẫn:

```
It also allows us to effectively enlarge the field of view of filters to
incorporate larger context without increasing the number of parameters or the
amount of computation.
```

Deeplab paper - phần abstract (https://arxiv.org/pdf/1606.00915.pdf)

Đối với tích chập austrous ta quan tâm tới các chỉ số:

- `rate` : Ký hiệu là R, là khoảng cách giữa các dòng hoặc cột liên tiếp nhau sau khi chèn thêm các dòng và cột 0. Nếu rate = r thì cứ cách một dòng hoặc cột là $r-1$ dòng hoặc cột 0.
- `kernel` : Ký hiệu K, kích thước bộ lọc.
- `padding` : Ký hiệu P, kích thước zero padding bao ngoài ảnh.
- `stride` : Ký hiệu S, số bước di chuyển của mỗi lượt tích chập.

Tính output shape cho Atrous Convolution

Gỉa sử áp dụng một tích chập Atrous Convolution (hoặc Dialition Convolution) với bộ lọc có kích thước K lên một ảnh input với kích thước W có padding P đều 2 phía và khoảng cách giãn cách giữa các dòng hoặc cột (rate) là R với bước di chuyển là S. Khi đó kích thước của output là:

$$W_{out} = \frac{(W-1)*R+1-(K-1)*R-1+2P}{S}+1 = \frac{(W-K)*R+2P}{S}+1$$

Việc chứng minh xin dành cho bạn đọc.

Theo công thức trên thì muốn tăng kích thước output ta có thể có 4 chiến lược là tăng R, tăng P, giảm K hoặc giảm S.

Ví dụ: Trong hình trên ở tích chập atrous với các thông số là
$W=3, K=3, P=2, S=1, R=2$ ta tính được kích thước output là:

$$W_{out} = \frac{(W-K)*R+2P}{S}+1 = \frac{(3-3)*2+2*2}{1}+1 = 5$$

muốn tăng kích thước lên k lần thì ta cần áp dụng:

$$kW = \frac{(W-K)*R+2P}{S}+1 \leftrightarrow R = \frac{(kW-1)S-2P}{W-K}$$

giả định $S=1$:

Top

$$R = \frac{(kW-1)-2P}{W-K}$$

Tích chập atrous trên tensorflow

Trên tensorflow chúng ta có thể tính tích chập atrous theo hai cách. Cách thứ nhất là dựa trên hàm tf.nn.atrous_conv2d() (https://www.tensorflow.org/api_docs/python/tf/nn/atrous_conv2d) và cách thứ 2 là dựa trên hàm tf.nn.conv2d() (https://www.tensorflow.org/api_docs/python/tf/nn/conv2d). Cả hai cách đều đưa ra cùng một kết quả. Để hiểu rõ hơn về cách tính những kết quả này, bạn đọc có thể xem thêm Atrous Convolution - Excel (https://docs.google.com/spreadsheets/d/17UcX6woX7cYqyVcftnZ6LSwNg6wdUmzRFTALzFBhjRY/edit?usp=sharing).

```
1       import numpy as np
2
3       x = np.array([[1, 0, 1, 0, 1],
4                     [0, 0, 0, 0, 0],
5                     [1, 0, 1, 0, 1],
6                     [0, 0, 0, 0, 0],
7                     [1, 0, 1, 0, 1]], dtype=np.float32)
8       x_pad = np.pad(x, pad_width=2)
9       x_in = x_pad.reshape((1, 9, 9, 1))
10      kernel = np.ones(shape=(3, 3, 1, 1), dtype=np.float32)
11      output = tf.nn.atrous_conv2d(x_in, kernel, rate=2, padding='VALID')
12      output
```

Top

```
<tf.Tensor: shape=(1, 5, 5, 1), dtype=float32, numpy=
array([[[[4.],
         [0.],
         [6.],
         [0.],
         [4.]],

        [[0.],
         [0.],
         [0.],
         [0.],
         [0.]],

        [[6.],
         [0.],
         [9.],
         [0.],
         [6.]],

        [[0.],
         [0.],
         [0.],
         [0.],
         [0.]],

        [[4.],
         [0.],
         [6.],
         [0.],
         [4.]]], dtype=float32)>
```

Theo cách thứ nhất chúng ta sẽ khai báo `rate = 2` . Lưu ý đầu vào của hàm là ma trận sau khi đã được padding và dilate.

```
x = np.array([[1, 0, 1, 0, 1],
              [0, 0, 0, 0, 0],
              [1, 0, 1, 0, 1],
              [0, 0, 0, 0, 0],
              [1, 0, 1, 0, 1]], dtype=np.float32)
x_pad = np.pad(x, pad_width=2)
x_in = x_pad.reshape((1, 9, 9, 1))
kernel = np.ones(shape=(3, 3, 1, 1), dtype=np.float32)

output = tf.nn.conv2d(x_in,
                      kernel,
                      strides=[1, 1],
                      padding="VALID",
                      dilations=[2, 2])
output
```

Top

```
<tf.Tensor: shape=(1, 5, 5, 1), dtype=float32, numpy=
array([[[[4.],
         [0.],
         [6.],
         [0.],
         [4.]],

        [[0.],
         [0.],
         [0.],
         [0.],
         [0.]],

        [[6.],
         [0.],
         [9.],
         [0.],
         [6.]],

        [[0.],
         [0.],
         [0.],
         [0.],
         [0.]],

        [[4.],
         [0.],
         [6.],
         [0.],
         [4.]]], dtype=float32)>
```

Theo cách thứ 2 chúng ta cần khai báo `dilation = 2` . Đầu vào tương tự như cách 1.

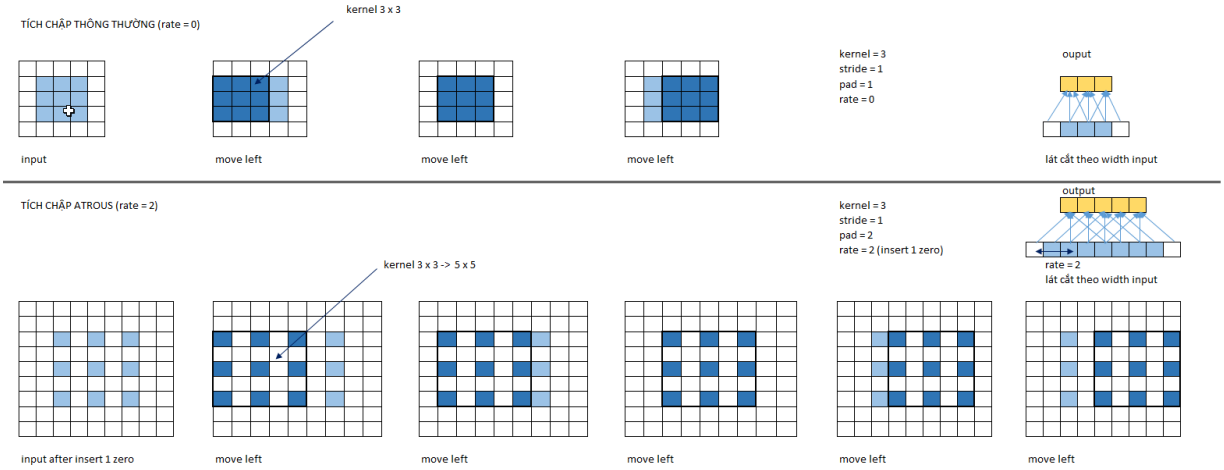
2.4. Atrous Spatial Pyramid Pooling (ASPP)

Trong DeepLabV1 và V2 để nhận diện được bối cảnh (context) của bức ảnh ở một vùng không gian rộng lớn hơn thì chúng ta phải điều chỉnh tăng `rate` của tích chập Atrous. Đồng thời giúp chúng ta có thể nhận diện các vật thể có thể xuất hiện trong ảnh với nhiều kích thước khác nhau.

Ví dụ: Nếu bạn đang muốn segment các vật thể trong một ảnh tham gia giao thông. Cùng là một chiếc xe ô tô nhưng nếu ở gần thì sẽ có kích thước lớn hơn và ở xa thì kích thước nhỏ hơn. Như vậy mô hình của chúng ta cần có cơ chế nhận dạng được đa dạng bối cảnh và kích thước.

ASPP (Atrous Spatial Pyramid Pooling) (Kim tự tháp bộ lọc atrous) là một cơ chế giúp ta thực hiện điều đó.

Top



Hình 3: Hình minh họa kiến trúc của một ASPP. ASPP là một tập hợp của nhiều bộ lọc atrous với kích thước khác nhau được thực hiện đồng thời trên cùng một vùng feature map (chính là `Input Feature Map` dòng dưới trong ảnh). Sự đa dạng về kích thước bộ lọc được xếp chồng lên nhau đã tạo thành một kim tự tháp bộ lọc atrous. Khi rate càng lớn, tầm nhìn của bộ lọc trên `Input Feature Map` càng lớn và giúp ta học được bối cảnh tổng quát (global context) tốt hơn. Với các bộ lọc kích thước nhỏ thì chúng ta sẽ học được bối cảnh cục bộ (local context) tốt hơn. Đồng thời ASPP cũng giúp phát hiện vật thể ở nhiều kích thước khác nhau. Sau cùng, các đặc trưng tổng quát và cục bộ được trộn lẫn với nhau để tạo thành Score map. Source: DeepLabV2 (https://arxiv.org/pdf/1606.00915.pdf)

2.5. Layer kết nối toàn bộ CRF

Layer kết nối toàn bộ CRF còn được gọi là Fully Connected CRF hoặc DenseCRF post-processing là một chu trình xử lý sau cùng để tính toán ra phân phối xác suất cho output. Layer kết nối toàn bộ CRF sẽ áp dụng kỹ thuật Conditional Random Field (viết tắt là CRF) để làm cho dự báo cho các pixel trở nên chuẩn xác hơn. Vậy CRF là gì?

2.5.1. Tác dụng của CRF

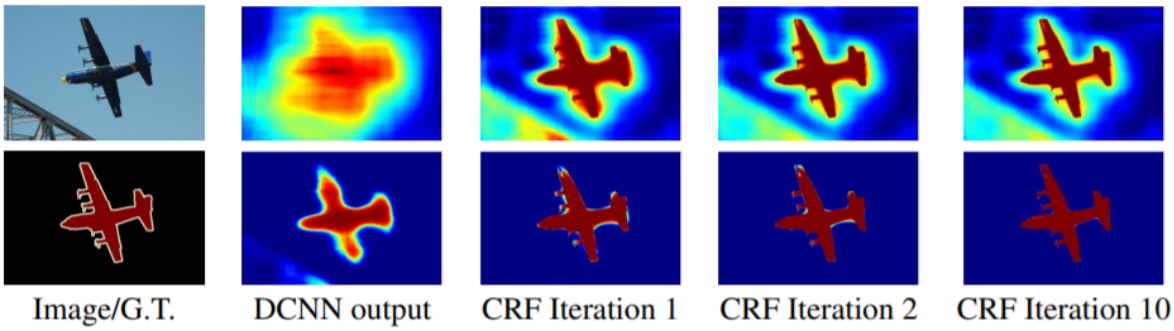
Chúng ta biết rằng không gian sẽ có tính liên kết. Dựa vào nội dung, bố cục, hình dạng và màu sắc của một vùng ảnh chúng ta có thể dự đoán được các vùng ảnh xung quanh. Ví dụ: bên cạnh một chiếc xe thường sẽ là những chiếc xe khác. Xét chi tiết hơn, nếu một pixel có nhãn và màu sắc xác định thì những pixels có khoảng cách **càng gần** với nó thì khả năng sẽ có **cùng nhãn và màu sắc**.

Đó chính là ý tưởng để áp dụng Conditional Random Field vào bài toán Image Segmentation để dự báo nhãn cho từng pixels.

Tóm tắt pipeline áp dụng CRF:

Sau khi sử dụng ASPP ta thu được một feature map tổng hợp bối cảnh ở nhiều vùng không gian có kích thước khác nhau. Sau đó để trở lại kích thước input, Feature map được scale up bằng bilinear interpolation. Các đường biên được tinh chỉnh rõ nét và hình dạng vật thể sẽ được làm cho smoothing hơn thông qua Fully Connected Conditional Random Field. Để hiểu rõ hơn CRF đã giúp cho dự báo cải thiện như thế nào, cùng quan sát hình bên dưới:

Top



Hình 4: Score map (đầu vào trước khi áp dụng hàm softmax) và belief map (đầu ra sau khi áp dụng hàm softmax). Chúng ta thể hiện score map (Dòng 1) và belief map (dòng 2) sau mỗi một lượt lấy `mean field` của mỗi vòng lặp CRF. Đầu ra của DCNN được sử dụng như là đầu vào cho quá trình `mean field`. Note: Mean field là một phương pháp xấp xỉ áp dụng trên các phân phối xác suất. Đi sâu vào khái niệm này rất rộng nên bạn đọc chỉ cần hiểu đơn giản như vậy. Nếu muốn hiểu về phương pháp này bạn đọc có thể tham khảo mean field approximation (http://bjlkeng.github.io/posts/variational-bayes-and-the-mean-field-approximation/). Source: DeepLabV2 (https://arxiv.org/pdf/1606.00915.pdf)

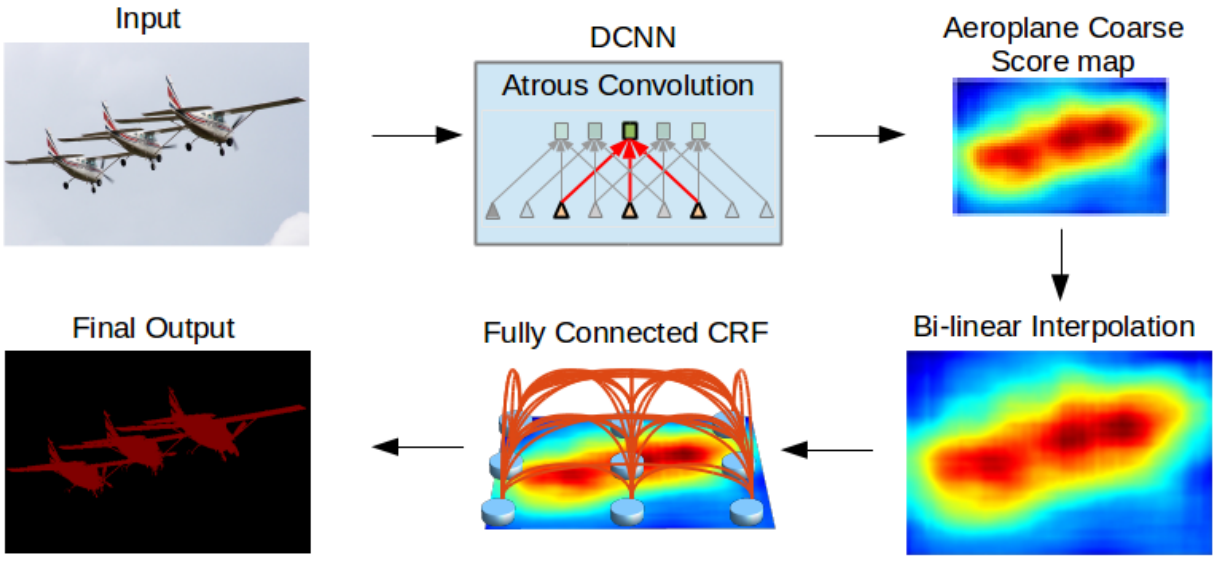
Như vậy ta thấy rằng sau các lượt áp dụng CRF liên tiếp thì hình ảnh dự báo đã trở nên smoothing hơn và đường biên rõ ràng hơn. Các vùng nhỏ có màu sắc khác nhau dần dần đã được làm mịn và đường biên không còn bị nhòe sáng. Bạn đọc đã hình dung ra tác dụng của CRF rồi chứ?

2.5.2. Phương pháp CRF

CRF là một phương pháp được sử dụng rộng rãi trong Image Segmentation nhằm kết hợp giữa điểm class được tính ra từ DCNN với thông tin bậc thấp (low-level feature) được ghi nhận bởi tương tác cục bộ giữa pixels với cạnh hoặc giữa các siêu pixels. Xin trích dẫn:

```
CRFs have been broadly used in semantic segmentation to combine class scores computed by multi-way classifiers with the low-level information captured by the local interactions of pixels and edges [23], [24] or superpixels [25]
```

Deeplab paper (https://arxiv.org/pdf/1606.00915.pdf)



Hình 5: Các layers của DeepLab model bao gồm: `input -> DCNN -> Score Map -> bilinear interpolation -> Fully Connected CRF -> mask output`.

Top

Thuật toán CRF sẽ tìm cách chuẩn hóa lại điểm số cho mỗi điểm ảnh thông qua một hàm năng lượng (Energy score) như sau:

$$\mathrm{E}(\mathbf{x}) = \sum_i \theta_i(x_i) + \sum_{ij} \theta_{ij}(x_i, x_j)$$

Ở đây $\mathbf{x}$ là nhãn được gán cho các pixels. Thành phần thứ nhất $\theta_i(x_i) = -\log P(x_i)$ là đối logratith xác suất của nhãn được gán cho pixel i và cũng chính là hàm `cross entropy` đánh giá chênh lệch giữa phân phối xác suất nhãn dự báo và thực tế.

Thành phần thứ 2 đánh giá khả năng ghép cặp tiềm năng giữa 2 pixels bất kỳ trong ảnh.

$$\theta_{ij}(x_i, x_j) = \mu(x_i, x_j) \left[w_1 \times \exp\left(-\frac{\|p_i - p_j\|^2}{2\sigma_\alpha^2} - \frac{\|I_i - I_j\|^2}{2\sigma_\beta^2}\right) + w_2 \times \exp\left(-\frac{\|p_i - p_j\|^2}{2\sigma_\gamma^2}\right) \right]$$

Ở đây $\mu(x_i, x_j) = 1$ nếu $x_i \neq x_j$ và bằng 0 nếu ngược lại. Nghĩa là chỉ các pixels khác nhãn thì mới bị phạt. Hàm $\exp(x) = e^x$ là hàm số mũ với cơ số tự nhiên e.

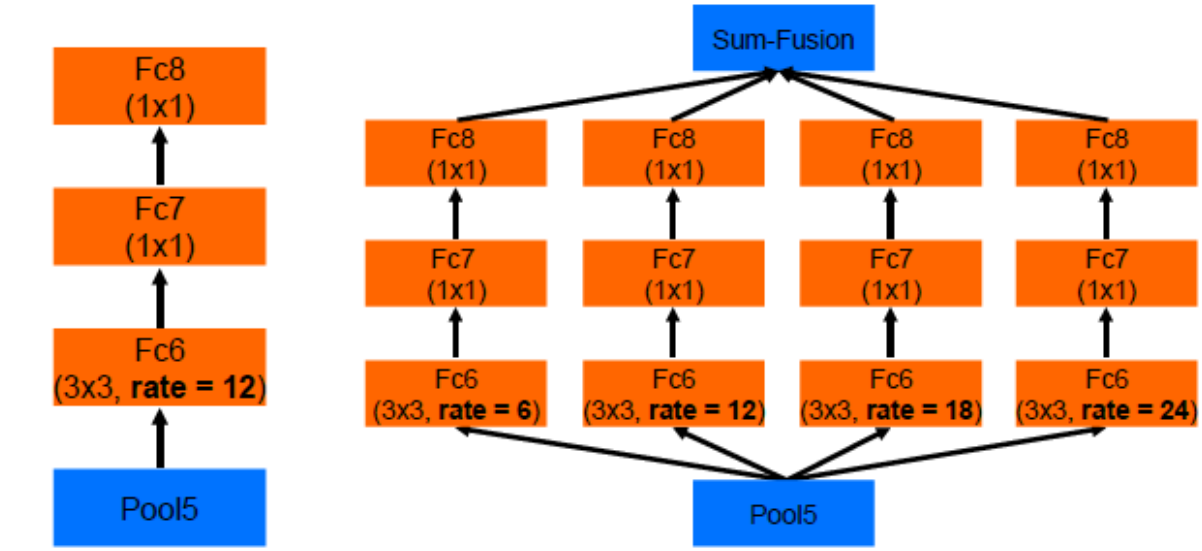
Bên trong ngoặc vuông là tổng có trọng số của 2 kernels trên 2 không gian khác nhau.

- Kernel đầu tiên $\exp\left(-\frac{\|p_i - p_j\|^2}{2\sigma_\alpha^2} - \frac{\|I_i - I_j\|^2}{2\sigma_\beta^2}\right)$ được gọi là `bilateral` kernel dựa trên chênh lệch về khoảng cách (kí hiệu là p) và chênh lệch về cường độ (ký hiệu là I). Kernel này buộc các pixels có cường độ và vị trí gần nhau thì sẽ có nhãn giống nhau và nó có mục đích chính là **bảo tồn cạnh** của vật thể.
- Kernel thứ hai $\exp\left(-\frac{\|p_i - p_j\|^2}{2\sigma_\gamma^2}\right)$ làm một `Gaussian Filter` chỉ dựa trên chênh lệch về khoảng cách. Cho phép xấp xỉ không gian và làm cho **hình ảnh trở nên smoothing hơn**. Nếu bạn đọc chưa biết về Gaussian Filter có thể xem lại Image Blur (https://phamdinhkhanh.github.io/2020/01/06/ImagePreprocessing.html#222-l%C3%A0m-m%E1%BB%9D-%E1%BA%A3nh-image-blurring).

Như vậy nhờ cơ thế bảo tồn cạnh và smoothing mà CRF đã cải thiện được kết quả dự báo.

2.6. Thực nghiệm DeepLab

Kiến trúc DeepLab sử dụng khá nhiều các điều chỉnh để tìm ra những điều chỉnh nào là tốt nhất theo phương pháp thử và học hỏi (`test and learn`).



(a) DeepLab-LargeFOV (b) DeepLab-ASPP

Top

Kiến trúc DeepLab-LargeFOV (bên trái: chỉ sử dụng một bộ lọc atrous), DeepLab-ASPP (bên phải: sử dụng một ASPP nhiều bộ lọc).

MSC	COCO	Aug	LargeFOV	ASPP	CRF	mIOU
						68.72
✓						71.27
✓	✓					73.28
✓	✓	✓				74.87
✓	✓	✓	✓			75.54
✓	✓	✓		✓		76.35
✓	✓	✓		✓	✓	77.69

Bảng kết quả tương ứng với mỗi điều chỉnh dựa trên backbone là mạng ResNet-101 và trên bộ dữ liệu PASCAL VOC 2012 Validation set.

- Đơn giản nhất sử dụng backbond ResNet-101: 68.72%
- MSC (Multiple Scale Input): Sử dụng đầu vào với nhiều kích thước.
- COCO (Models pretrained by COCO dataset): Sử dụng mô hình pretrained từ COCO dataset.
- Aug: Data augmentation bằng ngẫu nhiên scaling hình ảnh đầu vào với kích thước biến động từ 0.5 tới 1.5.
- LargeFOV: DeepLab sử dụng một bộ lọc Atrous duy nhất.
- ASPP: DeepLab sử dụng tích chập Atrous song song.
- CRF: Sử dụng Fully-connected CRF như là bước hậu xử lý sau cùng.

Cuối cùng, nó đã nhận được 77,69%. Và có thể thấy rằng MSC, COCO và Aug đóng góp sự cải thiện từ 68,72% lên 74,87%, điều này rất cần thiết với LargeFOV, ASPP và CRF.

3. DeepLabV3

Ở DeepLabV3 tác giả đưa vào 2 cải tiến chỉnh là:

- Tiến hành tích chập song song ASPP tại nhiều scale khác nhau và đưa thêm batch normalization, kế thừa ý tưởng từ mạng Inception (https://phamdinhkhanh.github.io/2020/05/31/CNNHistory.html#45-googlenet---inception-v3-2015).
- Đặc biệt, bỏ Fully Connected CRF tại bước xử lý sau cùng giúp gia tăng tốc độ tính toán.

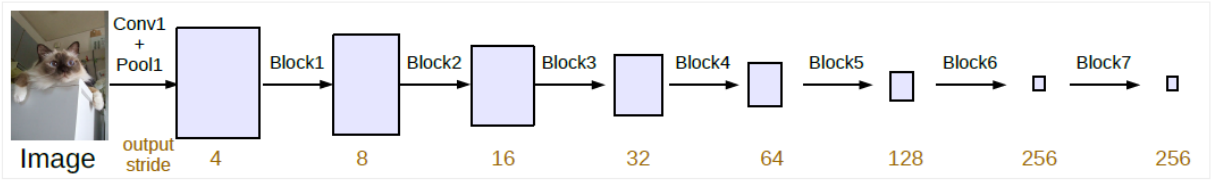
Một trong những thách thức của segmentation sử dụng mạng học sâu nhiều tầng DCNN là feature map ngày càng nhỏ hơn sau mỗi layer tích chập. Giảm độ phân giải có thể sẽ dẫn tới mất mát thông tin về vị trí và độ chi tiết của các đối tượng dự báo.

Chính vì vậy ở DeepLabV3 tác giả đã cố gắng điều chỉnh lại mức độ giảm độ phân giải ở các block trong mạng DCNN duy trì ở mức 16. Xin trích dẫn:

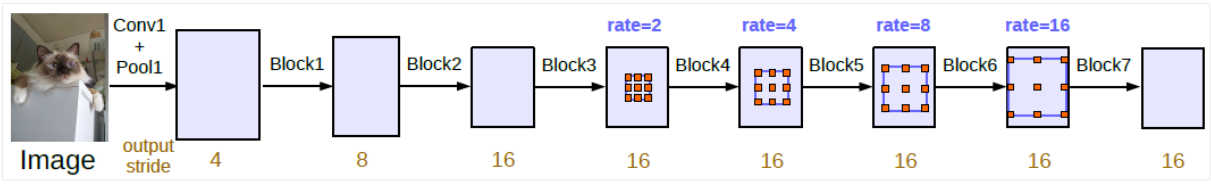
```
However, we discover that the consecutive striding is harmful for semantic
segmentation (see Tab. 1 in Sec. 4) since detail information is decimated, and
thus we apply atrous convolution with rates determined by the desired output
stride value, as shown in Fig. 3 (b) where output_stride = 16
```

Top

Source: DeepLabV3 - Rethinking Atrous Convolution for Semantic Image Segmentation (https://arxiv.org/pdf/1706.05587.pdf)



(a) Going deeper without atrous convolution.



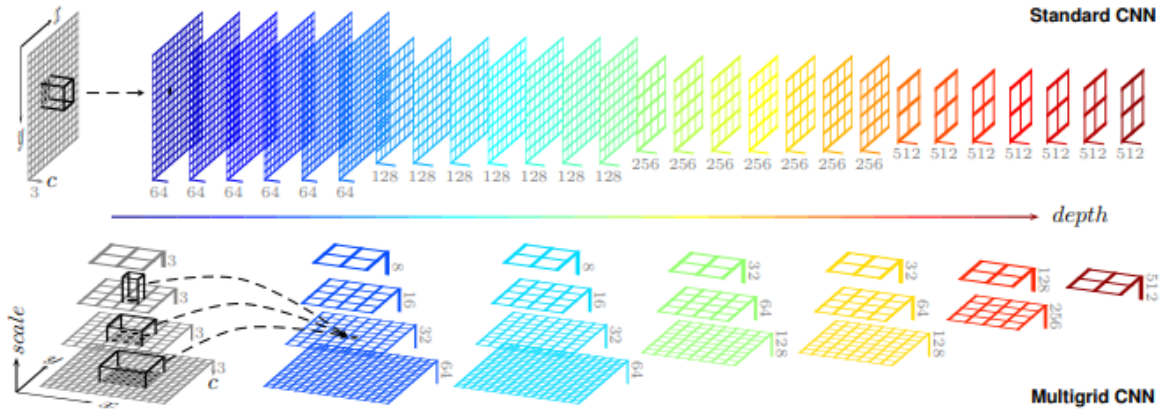
(b) Going deeper with atrous convolution. Atrous convolution with $rate > 1$ is applied after block3 when $output\_stride = 16$.

Hình 6: Độ phân giải của các block qua từng layer trong trường hợp áp dụng tích chập atrous (dòng trên) và không áp dụng tích chập atrous (dòng dưới). Khái niệm `output stride` trong hình là tỷ lệ độ phân giải của input so với độ phân giải feature map ở output. output stride càng tăng thì mức độ thu nhỏ của feature map càng lớn. Output stride đánh giá mức độ suy giảm tín hiệu mà input phải chịu khi truyền qua mạng.

Trong kiến trúc của DeepLabV3 tác giả áp dụng Atrous Convolution với đa dạng các kích thước như ở dòng dưới trong hình và tạo ra những cải thiện về `output stride` .

- **Không áp dụng Atrous Convolution**: Dòng đầu tiên, chúng ta chỉ áp dụng tích chập thông thường và max-pooling. Chúng ta thấy output stride gia tăng một cách đáng kể và khiến cho feature map nhỏ dần theo độ sâu của mô hình. Điều này gây hại cho segmentation bởi vì thông tin sẽ bị mất khi độ phân giải giảm tại những layers sâu hơn.
- **Áp dụng Atrous Convolution**: Dòng thứ 2, chúng ta có thể giữ cho độ phân giải của các block là ổn định và đồng thời gia tăng tầm nhìn (`field-of-view`) mà không cần gia tăng số lượng tham số và số lượng tính toán. Cuối cùng chúng ta thu được một feature map có kích thước lớn hơn và bảo toàn được thông tin về vị trí và không gian. Đây là một yếu tố có lợi cho segmentation.

3.1. Multi-Grid



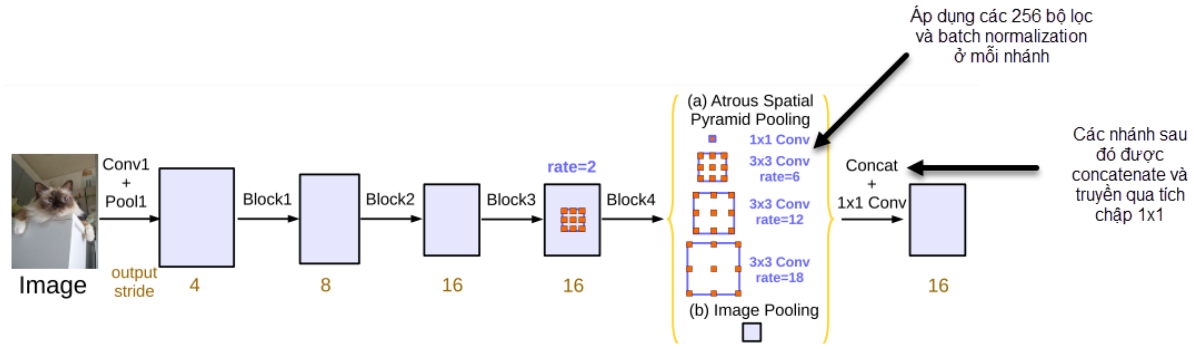
Hình 7: Kiến trúc multi-grid bên dưới và kiến trúc CNN chuẩn bên trên. Thay vì mỗi một layer chỉ áp dụng một bộ lọc duy nhất. Trong một mạng Multi-Grid, mỗi một layer có thể được huấn luyện trên một tập hơn bộ lọc có kích thước khác nhau dưới dạng kim tự tháp (pyramid). Chúng ta cần lựa chọn kích thước nào tương ứng ở mỗi pyramid level. Source: Multi-Grid Neural Network (https://arxiv.org/pdf/1611.07661.pdf)

Top

Trong DeepLabV3, tác giả đã kế thừa ý tưởng từ mạng đa lưới (Multi-Grid) sử dụng hệ thống phân cấp lưới (hierachy-grids) các tích chập với kích thước khác nhau. Tác giả định nghĩa Multi Grid = (r_1, r_2, r_3) là những unit rates cho 3 layers tích chập atrous ở mỗi block từ 4 đến 7. Cuối cùng, atrous rate ở mỗi tích chập atrous bằng tích của unit rate với rate tương ứng ở mỗi block. Ví dụ tại block 4 có `rate = 2`, khi `out_stride = 16` và `Multi_Grid = (1, 2, 4)`, ba tích chập atrous sẽ có `rates = 2 x (1, 2, 4) = (2, 4, 8)`. Tóm lại `atrous rates` cần áp dụng ở mỗi block sẽ bằng rate tương ứng ở mỗi block (như trong hình 6) nhân với `unit rates` của `Multi_Grid`. Để lựa chọn một cấu hình cho `atrous rates` cho mỗi một block từ 4 đến 7 thì tác giả lựa chọn cấu hình rates của multi-grid.

3.2. Atrous Spatial Pyramid Pooling

Tác giả cũng cải tiến ASPP bằng cách thêm batch normalization tại feature cuối cùng trước khi áp dụng Atrous Convolution. Sau đó kết quả được đưa qua một tích chập `1 x 1` với 256 bộ lọc. Sau cùng tác giả gia tăng độ phân giải thông qua Bilinear Upsampling để thu được kích thước mong muốn.



Hình 8: Module tiến hành song song của các các tích chập atrous. Source: DeepLabV3 (https://arxiv.org/abs/1706.05587)

Ví dụ trong hình 8 trả về cho chúng ta 2 outputs.

Output (a) là tích chập 3 x 3 với `Multi_Grid rate = (6, 12, 18)`.

Output (b) là đặc trưng ảnh. Sau đó các output này được concanate lại với nhau và truyền qua một tích chập kích thước `1 x 1`.

- ASPP đã được giới thiệu trong DeepLabv2. Ở thời điểm này, batch normalization (BN) từ Inception-V2 được thêm vào ASPP. Tác dụng của batch normalization đó là giúp cho mô hình hội tụ nhanh hơn.
- Lý do của việc sử dụng ASPP đó là thực tế cho thấy khi lấy mẫu rate lớn hơn, số lượng các bộ lọc hợp lệ giảm. Bộ lọc hợp lệ là bộ lọc có khả năng áp dụng cho các vùng feature hợp lệ, mà không phải padding thêm các giá trị 0. Sử dụng ASPP giúp làm đa dạng các bộ lọc với nhiều kích thước khác nhau và số lượng bộ lọc hợp lệ cũng nhiều hơn.
- Áp dụng một tích chập `1 x 1` và 3 tích chập `3 x 3` với atrous rates = $(6, 12, 18)$ khi output stride = 16.
- **Image pooling** hoặc **image-level feature** cũng được thêm vào để ghi nhận bối cảnh toàn bộ (global context). Image pooling sẽ được tạo ra bằng cách global average pooling của toàn bộ layer trước đó. Global context đã được chứng minh là giúp làm rõ hơn sự nhầm lẫn cục bộ (`local confusion`). Đây là ý tưởng được kế thừa từ ParseNet (https://arxiv.org/abs/1506.04579).

Top

- Ápp dụng 256 bộ lọc và batch normalization ở các nhánh của mỗi biến đổi trong ASPP tại các block từ 4 đến 7. Ý tưởng batch normalization kết thừa từ Inception.
- Kết quả các đặc trưng từ toàn bộ các nhánh được concatenate và truyền qua một tích chập `1 x 1` trước khi áp dụng tích chập `1 x 1` một lần nữa để tạo ra các giá trị logits từ hàm activation sigmoid.

3.3. Thực nghiệm

Để củng cố quan điểm của mình, tác giả đã so sánh giữa phương pháp xếp chồng (cascade) và Multi-Grid ResNets đối với ASPP. Kết quả cho thấy:

- **Tỷ lệ độ phân giải output/input** (Output Stride): Một độ phân giải lớn hơn, hoặc output stride nhỏ hơn, kết quả thể hiện là tốt hơn so với không áp dụng tích chập atrous hoặc output stride lớn hơn. Tác giả đồng thời cũng cho thấy rằng khi kiểm tra mạng trên tập validation của một output stride = 8 (độ phân giải cao hơn) thì kết quả tốt hơn so với một output stride = 16.
- **Xếp chồng** (Cascading): Kết quả cho thấy xếp chồng các tích chập atrous đã cải thiện so với tích chập thông thường. Tuy nhiên, tác giả cũng tìm thấy rằng càng nhiều block được thêm vào thì giá trị margin cải thiện càng nhỏ dần.
- **Multi-grid** (Mạng đa lưới): Kết quả của tác giả cho kiến trúc mạng đa lưới đã cải thiện tương đối so với một mạng `vanilla` và thể hiện kết quả tốt nhất khi $(r_1, r_2, r_3) = (1, 2, 1)$ tại block 7.
- **ASPP + Multi-Grid + Image Pooling**: Với Multi-grid rates tại $(r_1, r_2, r_3) = (1, 2, 4)$ tạo ra một ASPP(6, 8, 12). Mô hình đưa ra kết quả tốt nhất với 77.21% mIoU. Tại `output stride = 8` trên bộ dữ liệu COCO dataset với đa dạng kích thước đầu vào, model test với kết quả 82.70%, kết quả cải thiện lớn hơn so với sự thay đổi `output stride` từ 16 lên 8.

4. Kết luận

Như vậy ở bài này mình đã giới thiệu tới các bạn về nguyên lý hoạt động và kiến trúc của các mạng DeepLab ở các version 1, 2 và 3. Nhìn chung điểm mấu chốt trong kiến trúc của các mạng này vẫn là:

- Sử dụng tích chập Atrous để thu được tầm nhìn rộng hơn, giúp bảo toàn thông tin về không gian và độ chi tiết của các vật thể.
- Ở version 1 và 2, tác giả áp dụng CRF để tinh chỉnh nhãn đầu ra trên Score Map. Đây là một kỹ thuật dựa trên xác suất có điều kiện nhằm dự báo nhãn của pixels chuẩn xác hơn căn cứ vào khoảng cách vị trí giữa từng cặp pixels và cường độ pixels.
- Áp dụng ASPP, một tập hợp kim tự tháp của các bộ lọc atrous với các rates khác nhau. Giúp nhận diện được vật thể với đa dạng kích thước và trích xuất bối cảnh ở các tầm nhìn (field-of-view) khác nhau.
- Ở version 3 tác giả đã bỏ layer Fully Connected CRF ở cuối cùng ở bước hậu xử lý (post-processing) vì chi phí tính toán và thời gian dự báo của cao. Thay vào đó, tác giả đưa ra một kiến trúc Multi-Grid áp dụng nhiều bộ lọc atrous với rates khác nhau mà vẫn bảo toàn được độ phân giải thông qua kiểm soát output stride. Quá đó thông tin về không gian và độ chi tiết của ảnh đầu vào không bị mất đi. Kiến trúc xếp chồng (cascading) ở APSS cũng được thay thế bằng Multi-grid gồm các bộ lọc song song.

Top

Kỹ thuật xử lý của DeepLabV3 không quá phức tạp phải không nào? Thế nhưng kiến trúc này lại tạo ra được bất ngờ khi kết quả của nó khá tốt.

Method	mIOU
Adelaide_VeryDeep_FCN_VOC [85]	79.1
LRR_4x_ResNet-CRF [25]	79.3
DeepLabv2-CRF [11]	79.7
CentraleSupelec Deep G-CRF [8]	80.2
HikSeg_COCO [80]	81.4
SegModel [75]	81.8
Deep Layer Cascade (LC) [52]	82.7
TuSimple [84]	83.1
Large_Kernel_Matters [68]	83.6
Multipath-RefineNet [54]	84.2
ResNet-38_MS_COCO [86]	84.9
PSPNet [95]	85.4
IDW-CNN [83]	86.3
CASIA_IVA_SDN [23]	86.6
DIS [61]	86.8
DeepLabv3	85.7
DeepLabv3-JFT	86.9

Vì bài viết đã khá dài nên mình sẽ kết thúc tại đây. Ở bài sau mình sẽ hướng dẫn các bạn huấn luyện mô hình Image Segmentation trên DeepLabV3.

5. Tài liệu tham khảo

1. review Deeplabv1, Deeplabv2 (https://towardsdatascience.com/review-deeplabv1-deeplabv2-atrous-convolution-semantic-segmentation-b51c5fbde92d)
2. DeepLabV3 - Semantic Image Segmentation (https://towardsdatascience.com/deeplabv3-c5c749322ffa)
3. DeepLabV3 - Atrous convolution Semantic Segmentation (https://towardsdatascience.com/review-deeplabv3-atrous-convolution-semantic-segmentation-6d818bfd1d74)
4. DeepLab: Semantic Segmentation with DCNN, Atrous Convolution, and Fully Connected CRFs - Liang-Chieh Chen, George Papandreou,… (https://arxiv.org/pdf/1606.00915.pdf)
5. Rethinking Atrous Convolution for Semantic Image Segmentation - Liang-Chieh Chen, George Papandreou, Florian Schroff, Hartwig Adam (https://arxiv.org/abs/1706.05587)
6. Liang-Chieh Chen DeepLab Project (http://liangchiehchen.com/projects/DeepLab.html)

Top