Traditional image Segmentation 기술

Image segmenetation 모델에 적용된 객체 및 인식 및 위치 파악 기술은 특정 application이나 세그먼트화할 이미지 유형에 따라 달라진다.

우선 image Segmentation은 5가지로 나눠진다.(Threshold Method, Region Based Method, Edge Based Method, Watershed Based Method, Clustering Based Method)

#### 1. ****Threshold Method****

:임계값에 따라 이미지를 객체와 배경의 두 영역으로 나누는 기술이다. 이미지가 binary map으로 변환된다.

* 객체와 배경의 대비가 높은 이미지에 적합하다.
* image binarization에 자주 사용되므로 binary image에서만 작동하는 contour detection 및 identification과 같은 추가 알고리즘 사용 가능
* Otsu's Threshold, Mean shift 등이 있다.

#### 2. ****Region Based Method****

:인접한 픽셀 간의 유사성을 찾아 공통 클래스로 그룹화하여 작동한다.

* 일반적인 segmentatation procedure은 일부 pixels에 대하여 seed pixel로 설정하고, 해당 알고리즘은 seed pixel들의 immediate boundaries를 감지하여 similar or dissimilar를 분류
* immediate neighbors는 시드로 처리되고 전체 이미지가 segment화될 때까지 반복 ex. watershed algorithm
* **Watershed Based Method**가 euclidean distance map의 로컬 최대값에서 시작해 2개의 seed(시작 픽셀)가 동일한 지역 또는 segment에 속하는 것으로 분류될 수 없다는 제약 조건이 있다는 점에서 유사
* 다양한 특징을 가진 이미지에 적합
* 큰 이미지의 경우 느린 속도
* Region Growing, Region Split and Merge

#### 3. ****Edge Based Method****

:Edge detection. 이미지에서 어떤 픽셀이 edge 픽셀인지 분류하고 그에 따라 별도의 클래스에 따라 해당 edge 픽셀을 골라내는 작업

* 대비가 낮은 이미지에 적합
* 노이즈에 민감
* Canny, Gradient, Laplacia

#### 4. ****Watershed Based Method****

이미지에서 gradient의 크기로 구성된 gradient magnitude 영상을 구하고, 그 영상으로부터 Watershed(산의 봉우리, peak)를 구해 그 Watershed를 영역을 구분해주는 역할로 이용, 이미지를 segement화하는 방식

* 영상에 존재하는 detail이나 노이즈로 인해 gradient를 구했을 때 local minimus을 만들어내기 때문에 자연 영상에 대부분 over-segmentation 문제가 발생
* marker를 사용해 segmentation될 영역을 지정.(보통 최종적으로 segmentation되는 영역의 수 == marker의 수)
* Watershed, Marker Controlled Watershed

#### 5. ****Clustering Based Method****

: 최신 image segementation에서 일반적으로 가장 많이 쓰이는 것은 Clustering algorithm. 공통 속성을 가진 픽셀을 특정 segment에 속하는 것으로 함께 clustering하여 작동되는 unsupervised 알고리즘

* 시드 픽셀로 시작한 다음 유사성 기준에 따라 해당 픽셀을 중심으로 영역 확장
* 개체가 많은 이미지에 적합
* 계산 비용이 많이 들 수 있음
* K Means, Fuzzy C-Means

#### 6. ****Deep Learning Based Method****

: 머신러닝 알고리즘을 사용해 이미지에 있는 물체의 특징을 학습한 뒤, 알고리즘을 사용하여 세그먼트를 객체 또는 배경으로 분류

* 다양한 특징을 가진 이미지에 적합
* 많은 양의 학습 데이터 필요

Deep Learning Based Method

### **1. FCN(Fully Convolutional networks)**

* Semantic Segmentation의 대표적인 모델
* 기존 Classification용 CNN 모델의 문제
* AlexNet, VGG 등 분류에 자주 쓰이는 깊은 신경망들은 Semantic Segmentation에 부적합하다. 일반적으로 convolution 층들과 fully connected 층들로 이루어져 있으므로 항상 input image를 신경망에 맞는 고정된 사이즈로 작게 만들어 입력해줘야 한다.
* 분류용 CNN 모델들은 물체가 어떤 클래스에 속하는지 예측해낼 수 있으나 parmeter과 차원을 줄이는 layer를 갖고 있어 자세한 위치정보를 잃게될 수 있다.(마지막에 쓰이는 fully connected layer에 의해 물체가 어디에 존쟇는지 예측할 수 없게 됨)
* Pooling과 Fully connected layer를 없애고 stride와 padding을 1로 설정해 일정한 Convolution을 진행한다면 input의 차원은 보존할 수 있으나 parameter의 개수가 많아져 메모리 문제나 계산비용이 과다해진다.

**FCN 모델**

* Fully connected layer를 1x1 convolution 층으로 바꿈
* 네트워크 전체가 convolution 층들로 이루어지고, fully connected 층들이 없어졌으므로 입력 이미지 크기에 제한을 받지 않는다.
* 여러 층의 convolution 층들을 거치고 나면 feature map의 크기가 H/32xW/32가 되는데, 그 특성맵의 한 픽셀이 입력이미지의 32x32 크기를 대표  
  ➡ 입력이미지의 위치 정보를 '대략적으로' 유지
* convolution 층들을 거치고 나서 얻게 된 마지막 특성맵의 개수 == 훈련된 클래스의 개수
* 5개의 클래스로 훈련된 신경망이라면 5개의 특성맵(heatmap)을 산출
* 각 heatmap은 하나의 클래스를 대표
* 고양이 클래스에 대한 heatmap인 경우, 고양이가 있는 위치의 픽셀값들이 높음

**FCN 모델의 Architecture**

1. Convolution Layer를 통해 Feature 추출
2. 1x1 Convolution Layer를 통해, 낮은 해상도의 Class Presence Heat Map 추출
3. Transposed Convolution을 통해 낮은 해상도의 heatmap을 Upsampling한 뒤, input과 같은 크기의 Map 생성
4. Map의 각 pixel class에 따라 색칠한 뒤, Segmentation 결과 반환

* 1,2번 과정은 downsampling 단계로, convolution을 통해 차원을 축소한다.
* 3번 과정은 upsampling 단계로, 1,2번 과정을 통해 만들어진 heatmap의 크기를 원래 이미지의 크기로 다시 복원해주는 단계
  + 이미지의 모든 픽셀에 대해 클래스를 예측하는 것이 semantic segmentation의 목적이므로)
* 4번 과정은 upsampling된 heatmap을 종합해 최종적인 segmentation map을 만드는 단계
  + 각 픽셀당 확률이 가장 높은 클래스를 선정
  + 단순히 upsampling 시 원래 이미지 크기의 segmenation map을 얻을 수는 있으나 디테일하지 못함. 뭉뚱그려져 있고 디테일하지 못함. 한 번에 32배 upsampling하는 방법을 FCN-32s라고 함.
* Skip combining 기법  
  : convolution과 pooling 단계로 이루어진 이전 단계의 컨볼루션층들의 특성 맵을 참고하여 upsampling 시 좀더 정확도를 높일 수 있음(이전 컨볼루션층들의 특성맵들이 더 높은 해상도를 갖고 있으므로)
  + 전전 단계의 특성맵(pool3)과 전 단계의 특성맵(pool4)을 2배 upsampling한 것과 현 단계의 특성맵(conv7)d을 4배한 것을 모두 더한 다음에 (pool3 + 2xpool4 + 4xcon7)을 8배 upsampling을 해 얻은 특성맵들로 segmentation map을 얻는 방법을 FCN-8s라고 함. ➡ 일반적으로 사용되는 방법

### **2. U-Net**

* Semantic Segementation Task 수행에 널리 쓰이는 모델 중 하나로, 의학 이미지 Segmentation을 위해 개발된 U 형태의 모델이다.
* **U-Net의 장점**
  + 빠른 속도
    - 이미지를 인식하는 단위(Patch)에 대한 Overlap 비율이 적음
    - 기존의 모델에서 많이 사용되었던 Sliding Window 방식은 이전 Patch에서 검증이 끝난 부분을 다음 Patch에서 다시 검증하여 연산을 낭비하지만, U-Net에서는 이전 Patch에서 검증이 끝난 부분을 다음 Patch에서 중복하여 검증하지 않으므로 느린 연산 속도가 개선됨
  + Context와 Localization의 늪에서 탈출
    - Segmentation Network는 클래스 분류를 위한 인접 문맥 파악(Context)과 객체의 위치 판단(Localization)을 동시에 수행해야 한다.
    - 각 성능은 Patch의 크기에 영향을 받는데, 이 때 Trade-off(모순되는 관계)를 갖게 됨  
      ➡ **Patch의 크기가 커지면** 더 넓은 범위의 의미지를 한번에 인식할 수 있어 Context파악에는 탁월한 효과를 보이나, 많은 Max-Pooling을 거치며 Localization 성능이 저하됨  
      ➡ **Patch의 크기가 작아지면** Localization 성능은 좋아지나, 인식하는 범위가 지나치게 협소해져 Context 파악 성능이 저하됨  
      ➡ U-Net은 다층의 Layer의 Output을 동시에 검증해 이러한 모순적 관계를 극복함!
* **U-Net 모델의 Architecture**
  + 알파벳 U 형태 구조의 왼쪽 절반에 해당하는 Contracting Path와 오른쪽 절반에 해당하는 Expanding Path의 2가지로 분리됨
  + Contracting Path
    - **Encoder의 역할을 수행하는 부분으로, 전형적인 Convolution Network로 구성**
    - 입력을 Feature Map으로 변형해 이미지의 Context를 파악
    - 점진적으로 Spatial dimension을 줄여가며 고차원의 semantic 정보를 convolution filter가 추출해낼 수 있게 됨
    - Contracting Path의 앞단에 이미 잘 학습된 모델을 Backbone으로 사용해 학습 효율과 성능을 높일 수 있으며, 주로 ResNet 등의 모델을 사용함.
  + Expanding Path
    - **Decoder의 역할을 수행하는 부분으로, 전형적인 Upsampling + Convolution Network로 구성**
    - Convolution 연산을 거치기 전, Contracting Path에서 줄어든 사이즈를 다시 복원(Upsampling)하는 형태
    - Expanding Path에서는 Contracting을 통해 얻은 Feature Map을 Upsampling하고, 각 Expanding 단계에 대응되는 Contracting 단계에서의 Feature Map과 결합해서(Skip-Connection Concatenate) 더 정확한 Localization을 수행
    - encoder에서 spatial dimension 축소로 인해 손실된 spatial 정보를 점진적으로 복원하여 정교한 boundary segmentation을 완성
    - Multi-Scale Object Segmentation을 위해 DownSampling과 UpSampling을 순서대로 반복하는 구조
  + **U-Net이 다른 encoder-encoder-decoder 구조와 다른 점은 회색 선!**  
    ➡ Spatial 정보를 복원하는 과정에서 feature map 중 동일한 크기를 지닌 feature map을 가져와 prior로 활용  
    ➡ 더 정확한 boundary segmentation이 가능해짐

### **3. SegNet**

* U-Net과 마찬가지로 FCN의 구조를 이용한(Encoder-Decoder 유형의 구조) Image Segmentation 모델이다.
* Encoder에서는 VGG16의 13개 Convolution Layer를 동일하게 사용한다. Convolution, Pooling 구조들을 통해 Input Image의 feature map을 추출한다.
* Decoder에서는 Encoder에서 뽑은 feature map을 Upsampling과 Convolution하여 원래 크기로 되돌린다. Upsampling 후 마지막 Layer에서는 각 Class의 픽셀 단위 분류를 위한 Softmax Layer가 존재한다.
* **FCN, U-Net과의 차이점**
  + Max Pooling indices라는 정보를 활용한다는 점에서 차이점 발생!
  + Segnet은 Encoding 단계에서 pooling 시 'Max-Pooling indices' 정보를 기억한다.(Max-Pooling indices : pooling하기 전 data의 위치 정보)  
    Decoding 단계에서는 Encoding 과정에서 저장한 Max-Pooling indices를 이용하여 feature map을 upsampling한다.  
    ➡ **FCN**은 upsampling할 때 Deconvolution을 학습해야 하기 때문에 그만큼 학습을 위한 가중치 파라미터가 필요하지만, SegNet에서는 이 과정이 생략되기 때문에 학습 파라미터가 줄게 된다.  
    ➡ **U-Net**은 Decoding 과정에서 Skip combining하지만 U-Net은 Encoder의 같은 층 feature map 전체 정보를 Decoder로 전달하여 concat한다. 때문에 Max pooling indices의 일부 특징만 골라 사용하는 SegNet보다 무겁다.

### **4. DeepLab**

* 지금까지 version 1(2015), 2(2017), 3(2017), 3+(2018)가 출판된 semantic segmentation 방법으로, **Atrous convolution**을 적극적으로 활용할 것을 제안한다.
  + DeepLab V1 : atrous convolution 적용
  + DeepLab v2 : multi-scale context 적용하기 위한 Atrous Spatial Pyramid Pooling(ASPP) 기법 제안
  + DeepLab v3 : 기존 ResNet 구조에 atrous convolution을 활용해 좀더 dense한 feature map을 얻는 방법 제안
  + DeepLab v3+ : separable convolution과 atrous convolution을 결합한 atrous separable convolution의 활용 제안

**Atrous convolution**

* 기존 convolution과 달리 필터 내부에 빈 공간을 둔 채로 작동
* 얼마나 빈 공간을 둘지 결정하는 파라미터인 rate가 1일 때는 기존 convolution과 동일하고, rate가 커질수록 빈 공간이 넓어진다.
* 기존 convolution과 동일한 양의 파라미터와 계산량을 유지하면서 한 픽셀이 볼 수 있는 영역(field of view)를 크게 할 수 있게 된다.
  + 보통 semantic segmentation에서 높은 성능을 내기 위해서는, convolution neural network의 마지막에 존재하는 한 픽셀이 입력값에서 어느 크기의 영역을 커버할 수 있는지를 결정하는 receptive field의 크기가 중요하게 작용
  + Atrous convolution을 활용하면 파라미터 수를 늘리지 않으면서 receptive field를 크게 키울 수 있으므로 활용하면 좋음.

**Spatial Pyramid Pooling**

* Semantic segmentation의 성능을 높이기 위한 방법 중 하나로, spatial pyramid pooling 기법이 자주 활용됨
* DeepLab v2에서는 feature map으로부터, rate가 다른 여러 개의 atrous convolution을 병렬로 적용한 뒤, 이를 다시 합쳐주는 atrous spatial pyramid pooling(ASPP) 기법을 활용
  + - PSPNet에서도 atrous convolution을 활용하진 않지만 이와 유사한 pyramid pooling 기법을 적극 활용하고 있음
    - multi-scale context를 모델 구조로 구현하여 보다 정확한 semantic segmentation을 수행할 수 있도록 도움
* DeepLab v3+에서는 U-Net과 유사하게 [**intermediate connection**](https://velog.io/@hajieun02/Image-Segmentation2-%EA%B8%B0%EC%88%A0%EA%B3%BC-%EB%94%A5%EB%9F%AC%EB%8B%9D-%EA%B8%B0%EB%B0%98-%EB%AA%A8%EB%8D%B8#2-u-net)을 갖는 encoder-decoder 구조를 적용하여 보다 정교한 object boundary를 예측할 수 있게 됨

**Depthwise Separable Convolution**

* 입력 이미지가 8x8x3(HxWxC)이고, convolution filter 크기가 3x3(FxF)이라고 했을 때, filter 한 개가 가지는 파라미터 수는 3x3x3(FxFxC) = 27이 된다.
* 만약 filter가 4개 존재한다면, 해당 convolution의 총 파라미터 수는 3x3x3x4(FxFxCxN) = 108이 된다.
* 위 그림처럼 Convolution 연산에서 Channel 축을 filter가 한번에 연산하는 대신 아래 그림처럼 입력 영상의 Channel 축을 모두 분리시킨 뒤 filter의 channel 축 길이를 항상 1로 가지는 여러 개의 convolution filter로 대체시킨 연산을 **depthwise convolution**이라고 한다.

위의 depthwise convolution으로 나온 결과에 대해 1x1xC 크기의 convolution filter를 적용한 것을 **depthwise separable convolution**이라 한다.  
➡ 연산이 복잡하나, 기존의 convolution과 유사한 성능을 보이면서도 사용되는 파라미터 수와 연산량을 획기적으로 줄일 수 있음  
ex. 입력값이 8x8x3이고 16개의 3x3 convolution filter를 적용할 때 사용되는 파라미터의 개수는 **Convolution : 3x3x3x16 = 432 / Depthwise separable convolution : 3x3x3+3x17 = 75** 이다.  
➡ 기존 convolution filter가 spatial dimension과 channel dimension을 동시에 처리하던 것을 따로 분리시켜 각각 처리함. 여러 개의 필터가 spatial dimension 처리에 필요한 파라미터를 하나로 공유함으로써 파라미터의 수를 줄일 수 있게 되는 것!  
➡ 두 축을 분리시켜 연산을 수행하더라도 최종 결과값은 결국 두 가지 축 모두를 처리한 결과값을 얻을 수 있으므로, 기존 convolution filter가 수행하던 역할을 충분히 대체할 수 있음  
➡ **픽셀 각각에 대해 대해 label을 예측해야 하는 semantic segmentation을 위해 CNN 구조가 깊어지고, receptive field를 넓히기 위해 더 많은 파라미터를 사용하게 되는 상황에서, separable convolution을 활용하여 파라미터 수를 대폭 줄이면, 보다 깊은 구조로 확장해 성능 향상 및 메모리 사용량 감소/속도 향상 기대 가능**

* DeepLab V3+는 ResNet을 사용하는 encoder가 separable convolution을 적극 활용한 구조인 Xception으로 대체된다.
* Multi-scale context를 얻기 위해 활용되던 ASPP에는, separable convolution과 atrous convolution을 결합한 atrous separable convolution이 적용된 것으로 대체
* 기존에 단순하게 bilinear upsampling으로 해결했던 decoder 부분이 U-Net과 유사한 형태의 decoder로 대체
* Encoder와 ASPP, decoder 모두 separable convolution을 적극 활용함으로써 파라미터 사용량 대비 성능 효율을 극대화

### **5. ResNet**

* 마이크로소프트에서 개발한 알고리즘으로 'Deep Residual Learning for Image Recognition'을 의미한다.
* CNN 모델에서 층을 깊게 할 수록 Model Capacity가 크기 때문에 오버피팅이 발생하더라도 더 좋은 성능이 당연할 것이라 생각했으나 달라지지 않음.  
  ➡ Optimization의 문제. 네트워크가 더 깊어질수록 Optimize(Train)하는 것이 더 어렵기 때문에 Deep 네트워크가 Shallow 네트워크만의 퍼포먼스를 보이지 않은 것

**ResNet 모델의 Architecture**

* VCG-19의 구조를 뼈대로 하여 convolution layer를 추가해 깊게 만든 후, shortcut들을 추가
* plain 네트워크는 망이 깊어지면서 오히려 에러가 커졌으나, ResNet의 경우 망이 깊어지면서 에러가 작아졌다.  
  ➡ shortcut을 연결해 잔차(residual)를 최소화되도록 학습한 효과
* ➡ 152층의 ResNet이 가장 뛰어난 성능을 가짐.

**Residual Block**

입력값을 출력값에 더해줄 수 있도록 지름길을 하나 만들어준 것(잔차 연결)

* 동일한 연산을 하고 나서 input x를 더하는 것(Residual block, 기존에 학습한 정보를 보존, 추가적으로 학습하는 정보)과 더하지 않는 것(Plain layer, 기존에 학습한 정보를 보존하지 않고 변형시켜 새롭게 생성하는 정보)이 그 차이점
* Residual Connection(skip connection)을 통해 각각의 Layer(block)들이 작은 정보들을 추가적으로 학습하도록 함(=각각의 레이어가 배워야 할 정보량 축소)
* 기존의 신경망은 입력값 x를 타겟값 y로 매핑하는 함수 H(x)를 얻는 것이 목적이었으나, ResNet은 F(x)+x를 최소화하는 것을 목적으로 함.  
  ➡ F(x)가 0이 되면 출력과 입력이 모두 x로 같아지게 된다. F(x) = H(x)-x이므로 F(x)를 최소로 해준다는 것은 H(x)-x를 최소로 해주는 것과 동일한 의미를 지님. H(x)-x를 잔차라고 한다.  
  ➡ 잔차를 최소화 해주는 모델. ResNet

Residual Connection with Bottle Neck Layer

* Deep Network에서의 연산량을 줄이기 위해 GoogLeNet에서 사용했던 Bottle Neck Layer를 사용한 Residual Block를 사용함.
  + ResNet50 이상부터 BottleNeck을 사용한 Block을 사용함.
  + Input x의 dimension을 64로 축소한 후 f(x)를 연산하고 나중에 다시 256으로 복원시켜주는 것을 확인할 수 있다.