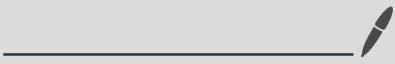
Semantic Image Segmentation With Deep Convolution Nets And Fully Connected CRFs



ABSTRACT

Deep Convolutional Neural Networks (DCNNs) have recently shown state of the art performance in high level vision tasks, such as image classification and object detection. This work brings together methods from DCNNs and probabilistic graphical models for addressing the task of pixel-level classification (also called "semantic image segmentation"). We show that responses at the final layer of DCNNs are not sufficiently localized for accurate object segmentation. This is due to the very invariance properties that make DCNNs good for high level tasks. We overcome this poor localization property of deep networks by combining the responses at the final DCNN layer with a fully connected Conditional Random Field (CRF). Qualitatively, our "DeepLab" system is able to localize segment boundaries at a level of accuracy which is beyond previous methods. Quantitatively, our method sets the new state-of-art at the PASCAL VOC-2012 semantic image segmentation task, reaching 71.6% IOU accuracy in the test set. We show how these results can be obtained efficiently: Careful network re-purposing and a novel application of the 'hole' algorithm from the wavelet community allow dense computation of neural net responses at 8 frames per second on a modern GPU.

픽셀단위로 분류를 하는 방법을 사용한다.

DCNN에서 기본에 사용하는 Segmantic Image Segmentation 방법이 정확도와 높은 증명함과 동시에 pixel 단위로 분류하기 위해서 새로운 "Deep Lab" system을 소개하려한다.

새롭게 사용한 기법은 두 — CRF

— Hole Algorithm 이 포함한다.

There are two technical hurdles in the application of DCNNs to image labeling tasks: signal downsampling, and spatial 'insensitivity' (invariance). The first problem relates to the reduction of signal resolution incurred by the repeated combination of max-pooling and downsampling ('striding') performed at every layer of standard DCNNs (Krizhevsky et al., 2013; Simonyan & Zisserman, 2014; Szegedy et al., 2014). Instead, as in Papandreou et al. (2014), we employ the 'atrous' (with holes) algorithm originally developed for efficiently computing the undecimated discrete wavelet transform (Mallat, 1999). This allows efficient dense computation of DCNN responses in a scheme substantially simpler than earlier solutions to this problem (Giusti et al., 2013; Sermanet et al., 2013).

The second problem relates to the fact that obtaining object-centric decisions from a classifier requires invariance to spatial transformations, inherently limiting the spatial accuracy of the DCNN model. We boost our model's ability to capture fine details by employing a fully-connected Conditional Random Field (CRF). Conditional Random Fields 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 (Rother et al., 2004; Shotton et al., 2009) or superpixels (Lucchi et al., 2011). Even though works of increased sophistication have been proposed to model the hierarchical dependency (He et al., 2004; Ladicky et al., 2009; Lempitsky et al., 2011) and/or high-order dependencies of segments (Delong et al., 2012; Gonfaus et al., 2010; Kohli et al., 2009; Chen et al., 2013; Wang et al., 2015), we use the fully connected pairwise CRF proposed by Krähenbühl & Koltun (2011) for its efficient computation, and ability to capture fine edge details while also catering for long range dependencies. That model was shown in Krähenbühl & Koltun (2011) to largely improve the performance of a boosting-based pixel-level classifier, and in our work we demonstrate that it leads to state-of-the-art results when coupled with a DCNN-based pixel-level classifier.

The three main advantages of our "DeepLab" system are (i) speed: by virtue of the 'atrous' algorithm, our dense DCNN operates at 8 fps, while Mean Field Inference for the fully-connected CRF requires 0.5 second, (ii) accuracy: we obtain state-of-the-art results on the PASCAL semantic segmentation challenge, outperforming the second-best approach of Mostajabi et al. (2014) by a margin of 7.2% and (iii) simplicity: our system is composed of a cascade of two fairly well-established modules, DCNNs and CRFs.

① DCNN의 문제점

— Signal downsampling

: 반복되는 pooling layer + downsampling에 의해서 네트워크가 깊어질수록 보존되는 정보가 너무 않다는 문제가 발생한다. 따라서 'atrous network'를 적용하였다.
이를 통해서 더 dense한 이미지의 구조와 특성을 파악할수 있게 되었다.

— Spatial insensitivity

: 더 넓은영역으로 시야는 넓혀서 확인 할수 있었다면, 범위 좁혔을것이다.

Moving towards works that lie closer to our approach, several other researchers have considered the use of convolutionally computed DCNN features for dense image labeling. Among the first have been Farabet et al. (2013) who apply DCNNs at multiple image resolutions and then employ a segmentation tree to smooth the prediction results; more recently, Hariharan et al. (2014a) propose to concatenate the computed inter-mediate feature maps within the DCNNs for pixel classification, and Dai et al. (2014) propose to pool the inter-mediate feature maps by region proposals. Even though these works still employ segmentation algorithms that are decoupled from the DCNN classifier's results, we believe it is advantageous that segmentation is only used at a later stage, avoiding the commitment to premature decisions.

More recently, the segmentation-free techniques of (Long et al., 2014; Eigen & Fergus, 2014) directly apply DCNNs to the whole image in a sliding window fashion, replacing the last fully connected layers of a DCNN by convolutional layers. In order to deal with the spatial localization issues outlined in the beginning of the introduction, Long et al. (2014) upsample and concatenate the scores from inter-mediate feature maps, while Eigen & Fergus (2014) refine the prediction result from coarse to fine by propagating the coarse results to another DCNN.

The main difference between our model and other state-of-the-art models is the combination of pixel-level CRFs and DCNN-based 'unary terms'. Focusing on the closest works in this direction, Cogswell et al. (2014) use CRFs as a proposal mechanism for a DCNN-based reranking system, while Farabet et al. (2013) treat superpixels as nodes for a local pairwise CRF and use graph-cuts for discrete inference; as such their results can be limited by errors in superpixel computations, while ignoring long-range superpixel dependencies. Our approach instead treats every pixel as a CRF node, exploits long-range dependencies, and uses CRF inference to directly optimize a DCNN-driven cost function. We note that mean field had been extensively studied for traditional image segmentation/edge detection tasks, *e.g.*, (Geiger & Girosi, 1991; Geiger & Yuille, 1991; Kokkinos et al., 2008), but recently Krähenbühl & Koltun (2011) showed that the inference can be very efficient for fully connected CRF and particularly effective in the context of semantic segmentation.

DCNN의 classifier을 만들기위해서 inter-mediate feature map를 픽셀 classification, 즉 segmation을 사용하려고 했으나 논문의 저자들은 segmentation은 나중단계에서 진행하는것이 섣부른 결정을 방지할수 있는것이라고 주장한다.

spatial localization이 DCNN에 문제중 하나로 언급이 되었는데, 기존의 연구는

① upsample and concatenate scores from inter-mediate feature maps

② refine the prediction result from coarse to fine by propagating coarse results to another DCNN

그러나 논문의 저자는은 다른 방법을 사용하고, 그 '다른 방법'이란

combination of pixel-level CRFs & DCNN based 'unary terms'이다.

모든 픽셀은 CRF node처럼 다루며, 따라서 CRF의 정보/계산은 그대로 이용해서 DCNN에 의해 계산되는 부분을 보정할수 있도록 한다.

Ⓐ DCNN (이경우에는 VGG-16)을 어떻게 변형시켜서 적용했는지 설명

Herein we describe how we have re-purposed and finetuned the publicly available Imagenet-pretrained state-of-art 16-layer classification network of (Simonyan & Zisserman, 2014) (VGG-16) into an efficient and effective dense feature extractor for our dense semantic image segmentation system.

VGG-16 model을 classification model, 즉 backbone으로 적용해서 해당 모델을 수정하는 방법으로 semantic image segmentation을 위한 모델을 완성시켰다.

3.1 EFFICIENT DENSE SLIDING WINDOW FEATURE EXTRACTION WITH THE HOLE ALGORITHM

Dense spatial score evaluation is instrumental in the success of our dense CNN feature extractor. As a first step to implement this, we convert the fully-connected layers of VGG-16 into convolutional ones and run the network in a convolutional fashion on the image at its original resolution. However this is not enough as it yields very sparsely computed detection scores (with a stride of 32 pixels). To compute scores more densely at our target stride of 8 pixels, we develop a variation of the method previously employed by Giusti et al. (2013); Sermanet et al. (2013). We skip subsampling after the last two max-pooling layers in the network of Simonyan & Zisserman (2014) and modify the convolutional filters in the layers that follow them by introducing zeros to increase their length ($2\times$ in the last three convolutional layers and $4\times$ in the first fully connected layer). We can implement this more efficiently by keeping the filters intact and instead sparsely sample the feature maps on which they are applied on using an input stride of 2 or 4 pixels, respectively. This approach, illustrated in Fig. 1 is known as the 'hole algorithm' ('atrous algorithm') and has been developed before for efficient computation of the undecimated wavelet transform (Mallat, 1999). We have implemented this within the Caffe framework (Jia et al., 2014) by adding to the *im2col* function (it converts multi-channel feature maps to vectorized patches) the option to sparsely sample the underlying feature map. This approach is generally applicable and allows us to efficiently compute dense CNN feature maps at any target subsampling rate without introducing any approximations.

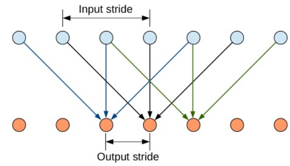


Figure 1: Illustration of the hole algorithm in 1-D, when *kernel_size = 3*, *input_stride = 2*, and *output_stride = 1*.

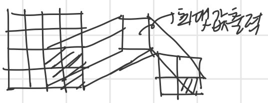
We finetune the model weights of the Imagenet-pretrained VGG-16 network to adapt it to the image classification task in a straightforward fashion, following the procedure of Long et al. (2014). We replace the 1000-way Imagenet classifier in the last layer of VGG-16 with a 21-way one. Our loss function is the sum of cross-entropy terms for each spatial position in the CNN output map (subsampled by 8 compared to the original image). All positions and labels are equally weighted in the overall loss function. Our targets are the ground truth labels (subsampled by 8). We optimize the objective function with respect to the weights at all network layers by the standard SGD procedure of Krizhevsky et al. (2013).

During testing, we need class score maps at the original image resolution. As illustrated in Figure 2 and further elaborated in Section 4.1, the class score maps (corresponding to log-probabilities) are quite smooth, which allows us to use simple bilinear interpolation to increase their resolution by a factor of 8 at a negligible computational cost. Note that the method of Long et al. (2014) does not use the hole algorithm and produces very coarse scores (subsampled by a factor of 32) at the CNN output. This forced them to use learned upsampling layers, significantly increasing the complexity and training time of their system: Fine-tuning our network on PASCAL VOC 2012 takes about 10 hours, while they report a training time of several days (both timings on a modern GPU).

optimizer = torch.optim.SGD(
param=[p for p in model.parameters()],
lr=0.005, momentum=0.9,
weight_decay=0.0005)

- pooling으로 인한 해상도 감소문제를 해결하는 segmentation에 이용되던 Fully Connected Layer의 경우에 Skip Connection이라는 방법을 사용해서 해결했었다.

pooling layer를 도입하는 이유는 Global-Feature, 즉 해당 객체의 두드러진 부분을 파악하기 위해서 인데 (Multi-scale로) 이는 사실 classification의 관점이다.

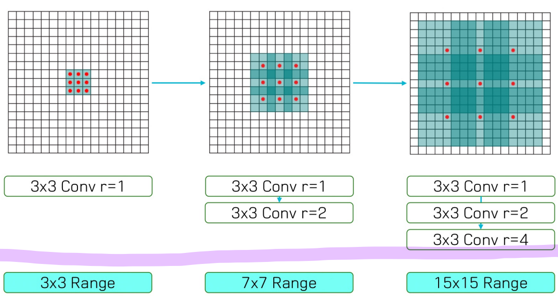


pooling을 도입하면 특성맵은 downsampling 되어서 특성맵의 크기는 줄게 된다.

- Dense prediction을 위해서는 up convolution과 multi scale input이 필수적이다.

의문 ① 굳이 downsampling을 한, 즉 굳이 pooling layer를 쓸 필요가 있는가?

의문 ② 굳이 여러 scale의 input을 변으로 처리해야 할까?

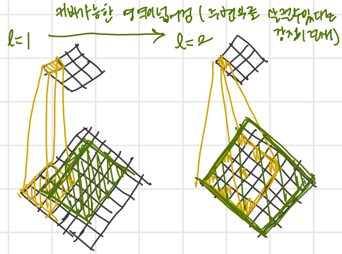


<Dilated Convolution>

$$(F *_l k)(p) = \sum_{s+lt=p} F(s)k(t)$$

- $l=1$: standard convolution
- $l>1$: dilated convolution.

rate의 크기를 늘릴수록 전체적으로 볼 수 있는 면적이 증가한다는 사실을 확인할 수 있다.



∴ 파라미터의 수는 유지하면서 pooling의 효과는 기대함과 동시에 resolution의 감소는 막는다.
(= downsampling)
↓
연산량을 줄이기 위해서

3.2 Controlling the Receptive Field Size and Accelerating Dense Computation with Convolutional Nets

Another key ingredient in re-purposing our network for dense score computation is explicitly controlling the network's receptive field size. Most recent DCNN-based image recognition methods rely on networks pre-trained on the Imagenet large-scale classification task. These networks typically have large receptive field size: in the case of the VGG-16 net we consider, its receptive field is 224×224 (with zero-padding) and 404×404 pixels if the net is applied convolutionally. After converting the network to a fully convolutional one, the first fully connected layer has 4,096 filters of large 7×7 spatial size and becomes the computational bottleneck in our dense score map computation.

cf) DCNN = Deep Convolutional Neural Network

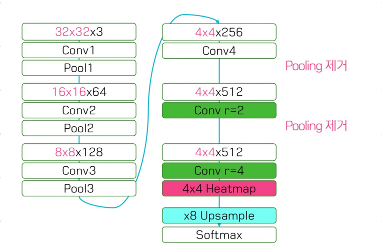
→ 갑작스럽게 네트워크상에서 데이터의 크기가 작은 부분

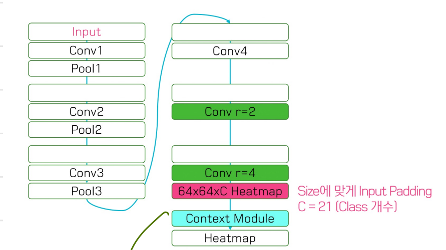
We have addressed this practical problem by spatially subsampling (by simple decimation) the first FC layer to 4×4 (or 3×3) spatial size. This has reduced the receptive field of the network down to 128×128 (with zero-padding) or 308×308 (in convolutional mode) and has reduced computation time for the first FC layer by $2-3$ times. Using our Caffe-based implementation and a Titan GPU, the resulting VGG-derived network is very efficient: Given a 306×306 input image, it produces 39×39

dense raw feature scores at the top of the network at a rate of about 8 frames/sec during testing. The speed during training is 3 frames/sec. We have also successfully experimented with reducing the number of channels at the fully connected layers from 4,096 down to 1,024, considerably further decreasing computation time and memory footprint without sacrificing performance, as detailed in Section 5. Using smaller networks such as Krizhevsky et al. (2013) could allow video-rate test-time dense feature computation even on light-weight GPUs.

- 원래는 DCNN에서 max-pooling layer 2개를 제거하는 방법을 사용함으로써 큰 크기의 featuremap을 얻고, atrous conv를 이용해 더 넓은 receptive field를 볼 수 있도록 했었다.
pooling을 적용한 뒤에 같은 크기의 kernel을 이용한 convolution layer를 적용하면 원래는 자연스럽게 receptive field가 넓어지는데, 여기서는 정확도를 위해서 해상도를 낮추는 pooling layer를 제거했기 때문에 atrous convolution을 이용해 pooling의 역할을 보완한다.

- 그러나 이런 과정을 단계적으로 수행해 나가는 과정에서 연산량이 많이 요구되고 있다. 때문에
첫번째 Fully Connected layer를 4x4 / 3x3의 spatial size로 줄였다.





VGG-16 network에서 pooling layer 2개를 제거한 뒤에 convolution layer에 마지막 2개에 artrous convolution을 적용

dilated convolution layer로만 구성된 새로운 network를 제안

4 Detailed Boundary Recovery: Fully-Connected Conditional Random Fields and Multi-scale Prediction

4.1 Deep Convolutional Networks and the Localization Challenge

As illustrated in Figure 2, DCNN score maps can reliably predict the presence and rough position of objects in an image but are less well suited for pin-pointing their exact outline. There is a natural trade-off between classification accuracy and localization accuracy with convolutional networks: Deeper models with multiple max-pooling layers have proven most successful in classification tasks, however their increased invariance and large receptive fields make the problem of inferring position from the scores at their top output levels more challenging.

→ 모델이 깊어질수록 분류의 정확성과 segmentation의 정확성을 위해 보강하기가 어렵다

Recent work has pursued two directions to address this localization challenge. The first approach is to harness information from multiple layers in the convolutional network in order to better estimate the object boundaries (Long et al., 2014; Eigen & Fergus, 2014). The second approach is to employ a super-pixel representation, essentially delegating the localization task to a low-level segmentation method. This route is followed by the very successful recent method of Mostajabi et al. (2014).

In Section 4.2, we pursue a novel alternative direction based on coupling the recognition capacity of DCNNs and the fine-grained localization accuracy of fully connected CRFs and show that it is remarkably successful in addressing the localization challenge, producing accurate semantic segmentation results and recovering object boundaries at a level of detail that is well beyond the reach of existing methods.

B) 더 세밀한 boundary를 구축해 위한 Localization Challenge를 극복한 방법

— 논문저자들은 Fully Connected CRF를 적용해서 이 문제를 해결하고자 한다.

To overcome these limitations of short-range CRFs, we integrate into our system the fully connected CRF model of Krähenbühl & Koltun (2011). The model employs the energy function

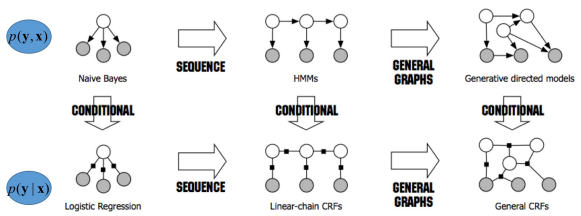
$$E(\boldsymbol{x}) = \sum_i \theta_i(x_i) + \sum_{ij} \theta_{ij}(x_i, x_j) \tag{1}$$

where $\boldsymbol{x}$ is the label assignment for pixels. We use as unary potential $\theta_i(x_i) = -\log P(x_i)$, where $P(x_i)$ is the label assignment probability at pixel i as computed by DCNN. The pairwise potential is $\theta_{ij}(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^{K} w_m \cdot k^m(\boldsymbol{f}_i, \boldsymbol{f}_j)$, where $\mu(x_i, x_j) = 1$ if $x_i \neq x_j$, and zero otherwise (*i.e.*, Potts Model). There is one pairwise term for each pair of pixels i and j in the image no matter how far from each other they lie, *i.e.* the model's factor graph is fully connected. Each k^m is the Gaussian kernel depends on features (denoted as $\boldsymbol{f}$) extracted for pixel i and j and is weighted by parameter w_m. We adopt bilateral position and color terms, specifically, the kernels are

$$w_1 \exp\Big(-\frac{||p_i - p_j||^2}{2\sigma_\alpha^2} - \frac{||I_i - I_j||^2}{2\sigma_\beta^2}\Big) + w_2 \exp\Big(-\frac{||p_i - p_j||^2}{2\sigma_\gamma^2}\Big) \tag{2}$$

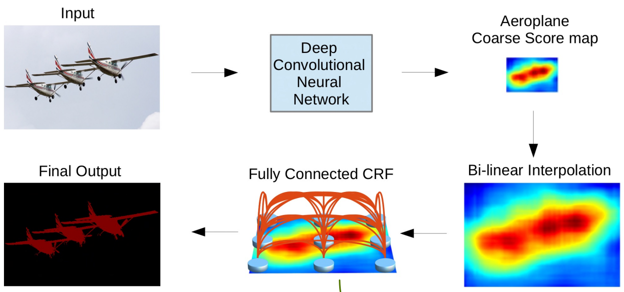
where the first kernel depends on both pixel positions (denoted as p) and pixel color intensities (denoted as I), and the second kernel only depends on pixel positions. The hyper parameters $\sigma_\alpha, \sigma_\beta$ and σ_γ control the "scale" of the Gaussian kernels.

Crucially, this model is amenable to efficient approximate probabilistic inference (Krähenbühl & Koltun, 2011). The message passing updates under a fully decomposable mean field approximation $b(\boldsymbol{x}) = \prod_i b_i(x_i)$ can be expressed as convolutions with a Gaussian kernel in feature space. High-dimensional filtering algorithms (Adams et al., 2010) significantly speed-up this computation resulting in an algorithm that is very fast in practice, less than 0.5 sec on average for Pascal VOC images using the publicly available implementation of (Krähenbühl & Koltun, 2011).



CRF

- Graph Model
- 각 node = label of pixel
- 각 node의 latent variable = pixel (예컨대 RGB값)
- 각 node 마다 확률 Modeling
- posterior은 최대화하므로 확률 Model 학습



→ 모든 노드 즉 픽셀들이 서로 연결되어서
기존의 CRF보다 밀접된 계산을 하게 된다.
다만 계산량이 많기 때문에 MeanField Approximation을 적용.

Fully Connected CRFs

Maximize Posterior

$$P(X|I) = \frac{1}{Z(I)} \exp\left(-\sum \phi_c(X_c|I)\right)$$

(X: label, I: Image, $Z(I)$: Normalization)

Minimize Energy
(=확률을 최대화하기 위한 보조)

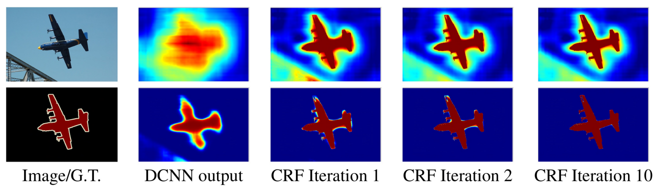
$$E(X|I) = \sum \phi_c(X_c|I)$$

$$E(X) = \sum \psi_c(X_c)$$

$$E(X) = \sum_i \psi_i(x_i) + \sum_{i,j} \psi_{i,j}(x_i, x_j)$$

(Unary, Fully Connected)

$$= \sum_i -\log P(x_i) + \sum_{i,j} \psi_{i,j}(x_i, x_j)$$



↳ CRF를 적용해 테두리의 정확도가 상승했음을 확인할수 있다.

Pairwise Term (Impose consistency of labeling) → defined over neighboring pixels

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

$$\mu(x_i, x_j) = \begin{cases} 1 & (x_i \neq x_j) \\ 0 & (\text{else}) \end{cases}$$

∴ 픽셀이 서로 비슷한 이미지인데 (위치정보나 RGB 정보가)
label값이 서로 다르면 energy가 증가하여 penalty를 받게 한다.

* p_i, p_j: 픽셀의 위치
* I_i, I_j: 픽셀의 RGB값
* w_1, w_2: kernel weights
* $\sigma_\alpha, \sigma_\beta, \sigma_\gamma$: Hyper-parameter

cf) 여기서 kernel이라는것의 개념은결국 convolution, 합성곱 연산을 할때 각 특징에부여 할 숫자들이 적혀있는 이미지데이터를 계속 훑고 지나가는 map이라 생각.

그러나 계산량이 많기 때문에 Mean Field Approximation을 사용한다.

$$P(X|I) = \frac{1}{Z(I)} \exp(-E(X))$$

대신 $Q(X) = \prod_i Q_i(X_i)$를 정의하고 $D_{KL}(Q||P)$를 최소화하도록 하였다.

$$Q_i(x_i = l) = \frac{1}{Z_i} \exp\left(-\psi_u(x_i) - \sum_{l'} \mu(l, l') \sum_m w^m \sum_j k^m(f_i, f_j) Q_j(l')\right)$$

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

→ appearance kernel (first term)

→ Smoothness kernel (second term)

Update Rule

$$Q_i(x_i = l) = \frac{1}{Z_i} \exp\left(-\psi_u(x_i) - \sum_{l'} \mu(l, l') \sum_m w^m \sum_j k^m(f_i, f_j) Q_j(l')\right)$$

→ MeanField Update rule for CRF

초기화) $Q_i(x_i = l) = \frac{1}{Z_i} \exp(-\psi_u(x_i))$

수렴할때까지...

$$\tilde{Q}_i^m(l) = \sum_j k^m(f_i, f_j) Q_j(l)$$ Message Passing

(k^m → Gaussian Kernel. f_i, f_j → feature vectors for pixel i, j)

$$\check{Q}_i(l) = \sum_m w^m \tilde{Q}_i^m(l)$$ Weighing (가중치부여)

(w^m → weight of the mth kernel)

$$\hat{Q}_i(l) = \sum_{l'} \mu(l, l') \check{Q}_i(l)$$ Compatibility Transform

($\mu(l, l')$ → label compatibility function)

$$\acute{Q}_i(l) = -\left(\psi_u(x_i) + \hat{Q}_i(l)\right)$$ Adding Unary (Local Update)

$$Q_i = \frac{1}{Z_i} \exp(\acute{Q}_i(l))$$ Normalization (softmax)