University of Illinois Fall 2018

ECE 544NA: Pattern Recognition
Lecture 14 : October 16

Lecturer: Alexander Schwing Scribe: Yuqi Li

1 Motivation: Structured Learning

Figure 1: Image segmentation problem, $|\mathcal{Y}| = K^{\#\text{ pixels}}$

Our general setting of learning: a mapping from inputs $x \in \mathcal{X}$ to one dimensional discrete $y \in \mathcal{Y} = \{1, ..., K\}$, may be insufficient when dealing with high dimensional structured objects. For example, in image segmentation problem, fig.(1), the output y are segmentation labels of every pixels in the input image. Predicting every single pixel's label from an image, can be achieved, provided that we have infinite computation resources. But it requires N independent trained classifiers, where N is the number of pixels. However, if we consider grouping all the labels as an structured object, some part of classifier could be reused to save computation amount. Also, the correlation between coordinates of image could contribute to better performance in classification. Thus, we need to study on structured learning.

2 Review on Structured Prediction

In the previous lecture (L12,L13) on structured prediction, we tried to solve the inference step in classification task in higher dimension:

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} F(\mathbf{w}, x, \mathbf{y}), \mathbf{y} \in \{1, ..., K\}^D \tag{1}$$

The exponentially growing search space becomes an issue when the goal of learning changed from a single dimension $y^{(i)} \in \{1, ..., K\}$ (e.g. a class label of images) to $\mathbf{y}^{(i)} \in \{1, ..., K\}^D$ (e.g. segmentation labels of every pixel in images). To simplify the search space, scoring function was decomposed

to summation of scoring function of a restricted set $r \in \mathcal{R} \triangleq \mathcal{P}(\{1, ..., K\})$ (power set)

$$F(\mathbf{w}, x, \mathbf{y}) = \sum_{r \in \mathcal{R}} f_r(\mathbf{w}, x, \mathbf{y}_r) \tag{2}$$

Under such assumption, the dimension of search space reduces from K^D to $\sum_{r \in \mathcal{R}} K^{|r|}$.

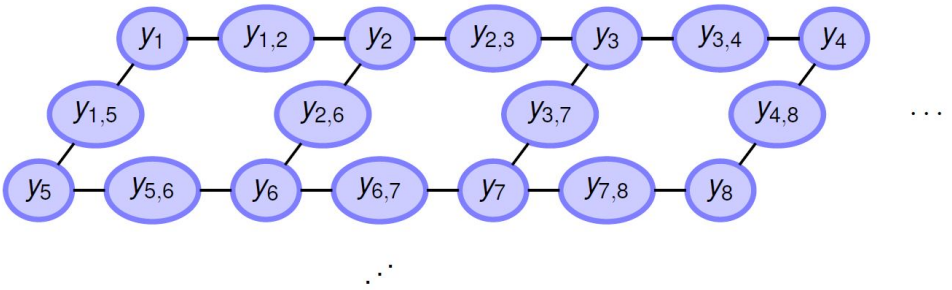

Figure 2: Graph node of unary and pairwise terms

Each $f_r(\mathbf{w}, x, \mathbf{y}_r)$ corresponds to a node in a graph model. For example (fig:2), in image segmentation problem, a node could represent unary term ($|r| = 1$) like the image evidence of a single pixel, or pairwise term ($|r| = 2$) that captures the correlation between adjacent pixels and encourages smoothness, or higher order terms if the graph model of output becomes more complex.
Now the optimization problem can be rewritten as:

$$\mathbf{y}^* = \arg\max_{\hat{y}} \sum_{r \in \mathcal{R}} f_r(\mathbf{w}, x, \mathbf{y}_r) \tag{3}$$

The following inference algorithms were introduced in last two lectures to solve for a inference optimization problem: dynamic programming, integer linear program, linear programming relaxation, message passing, graph-cut.

3 Linear Scoring Function

Now in the context of image segmentation, the easiest way to combine the unary terms and pairwise terms is to replace the scoring function with the **linear combination** of terms.

$$\begin{aligned} F(\mathbf{w}, x, \mathbf{y}) &= \mathbf{w}^T f(x^{(i)}, \mathbf{y}) \\ &= w_1 \underbrace{\sum_{r \in \mathcal{R}, |r|=1} f_r(x^{(i)}, \mathbf{y}_r)}_{\text{unary term}} + w_2 \underbrace{\sum_{r \in \mathcal{R}, |r|=2} f_r(x^{(i)}, \mathbf{y}_r)}_{\text{pairwise term}} \end{aligned} \tag{4}$$

If we have complex models consisting of more than unary and pairwise terms, we may expand the scoring function to M graphical models:

$$f(x^{(i)}, \mathbf{y}) = \begin{bmatrix} f_1(x^{(i)}, \mathbf{y}) \\ ... \\ f_M(x^{(i)}, \mathbf{y}) \end{bmatrix} = \begin{bmatrix} \sum_{r \in \mathcal{R}_1} f_{1,r}(x^{(i)}, \mathbf{y}_r) \\ ... \\ \sum_{r \in \mathcal{R}_M} f_{M,r}(x^{(i)}, \mathbf{y}_r) \end{bmatrix} \tag{5}$$

$$F(\mathbf{w}, x, \mathbf{y}) = \mathbf{w}^T f(x^{(i)}, \mathbf{y}) = \sum_{m=1}^{M} w_m \underbrace{\Big(\sum_{r \in \mathcal{R}_m} f_{m,r}(x^{(i)}, \mathbf{y}_r) \Big)}_{\text{individual models}} = \underbrace{\sum_{r \in \mathcal{R}} \hat{f}_r(\mathbf{w}, x^{(i)}, \mathbf{y}_r)}_{\text{combined models}} \tag{6}$$

where each $\mathcal{R}_m$ represents the nodes in mth graphical model, and w_m represents the weight of mth graphical model. Here the combined models is defined on the whole graph, and the weight of each graphical model is embedded inside $\hat{f}_r(\mathbf{w}, x^{(i)}, \mathbf{y}_r)$

4 Structured SVM

Given this linear scoring function form with weights $\mathbf{w}$, how to express the learning objective and learn $\mathbf{w}$? From the previous lecture on multiclass SVM, we have the objective function as:

$$\min_{\mathbf{w}} L(\mathbf{w}) \triangleq \frac{C}{2}\|\mathbf{w}\|_2^2 + \sum_{i \in \mathcal{D}} \left(\underbrace{\max_{\hat{\mathbf{y}}} \Big(\mathbf{w}^T f(x^{(i)}, \hat{\mathbf{y}}) + L(\hat{\mathbf{y}}, \mathbf{y}^{(i)}) \Big)}_{\text{Loss-augmented inference}} - \mathbf{w}^T f(x^{(i)}, \mathbf{y}^{(i)}) \right) \tag{7}$$

Given previous efficient algorithms on structured inference, we have the iterative algorithm to find the weights as follows:

1. Loss-augmented inference

$$\mathbf{y}^* = \arg\max_{\hat{\mathbf{y}}} \Big(\mathbf{w}^T f(x^{(i)}, \hat{\mathbf{y}}) + L(\hat{\mathbf{y}}, \mathbf{y}^{(i)}) \Big) \tag{8}$$

2. Perform gradient step:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \nabla_{\mathbf{w}} L(\mathbf{w}) \tag{9}$$

Since this is still a linear objective function with respect to $\mathbf{w}$, the gradient is:

$$\nabla_{\mathbf{w}} L(\mathbf{w}) = C\mathbf{w} + \sum_{i \in \mathcal{D}} \big(f(x^{(i)}, \mathbf{y}^*) - f(x^{(i)}, \mathbf{y}^{(i)}) \big) \tag{10}$$

It is worth noting that in each iteration, we need to solve one structured prediction task per sample per iteration. And we need solver that is efficient and accurate enough to perform the inference step independent of weight update.

$$\min_{\mathbf{w}} \frac{C}{2}\|\mathbf{w}\|_2^2 + \sum_{i \in \mathcal{D}} \epsilon \ln \sum_{\hat{y}} \exp \frac{L(\mathbf{y}^{(i)}, \hat{\mathbf{y}}) + \mathbf{w}^T f(x^{(i)}, \hat{\mathbf{y}})}{\epsilon} - \mathbf{w}^T f(x^{(i)}, \mathbf{y}^{(i)}) \tag{11}$$

Similar to structured SVM, structured multiclass logistic regression could be formulated using the general classification framework (11), the iterative algorithms follows the same principle.

5 Deep Structured Learning

In the previous section, the scoring function is set to be linear with $\mathbf{w}$. In the scenario of single label prediction, in order to achieve better classification accuracy, we turned to kernel methods and

deep network models, introducing a nonlinear scoring function. This idea also applies to structured learning.

$$\min_{\mathbf{w}} L(\mathbf{w}) \triangleq \frac{C}{2}\|\mathbf{w}\|_2^2 + \sum_{i\in\mathcal{D}} \epsilon \ln \sum_{\hat{y}} \exp \frac{L(\mathbf{y}^{(i)}, \hat{\mathbf{y}}) + F(\mathbf{w}, x^{(i)}, \hat{\mathbf{y}})}{\epsilon} - F(\mathbf{w}, x^{(i)}, \mathbf{y}^{(i)}) \tag{12}$$

With the objective function above, we can write down the gradient descent step ($\epsilon = 1, L(\mathbf{y}^{(i)}, \hat{\mathbf{y}}) = 0$ for simplicity)

$$\begin{aligned} p(\hat{\mathbf{y}}; x, \mathbf{w}) &\triangleq \frac{\exp F(\mathbf{w}, x^{(i)}, \hat{\mathbf{y}})}{\sum_{\hat{\mathbf{y}}} \exp F(\mathbf{w}, x^{(i)}, \hat{\mathbf{y}})} \\ \nabla_{\mathbf{w}} L(\mathbf{w}) &= \nabla_{\mathbf{w}} \left(\frac{C}{2}\|\mathbf{w}\|_2^2 + \sum_{i\in\mathcal{D}} \ln \sum_{\hat{y}} \exp F(\mathbf{w}, x^{(i)}, \hat{\mathbf{y}}) - F(\mathbf{w}, x^{(i)}, \mathbf{y}^{(i)}) \right) \\ &= C\mathbf{w} + \sum_{i\in\mathcal{D}} \Big(\sum_{\hat{\mathbf{y}}} p(\hat{\mathbf{y}}; x, \mathbf{w}) \nabla_{\mathbf{w}} F(\mathbf{w}, x^{(i)}, \hat{\mathbf{y}}) - \nabla_{\mathbf{w}} F(\mathbf{w}, x^{(i)}, \mathbf{y}^{(i)}) \Big) \\ &= C\mathbf{w} + \sum_{i\in\mathcal{D}} \sum_{\hat{\mathbf{y}}} \Big(p(\hat{\mathbf{y}}; x, \mathbf{w}) - \delta(\hat{\mathbf{y}} - \mathbf{y}^{(i)}) \Big) \nabla_{\mathbf{w}} F(\mathbf{w}, x^{(i)}, \hat{\mathbf{y}}) \end{aligned} \tag{13}$$

5.1 Deep structured learning algorithm

Each iteration in this optimization process operates as follows:

1. Forward pass to compute $F(\mathbf{w}, x^{(i)}, \hat{\mathbf{y}})$
2. Compute $p(\hat{\mathbf{y}}; x, \mathbf{w})$ via soft-max
3. Backward pass via chain rule to obtain gradient
4. Update parameters $\mathbf{w}$

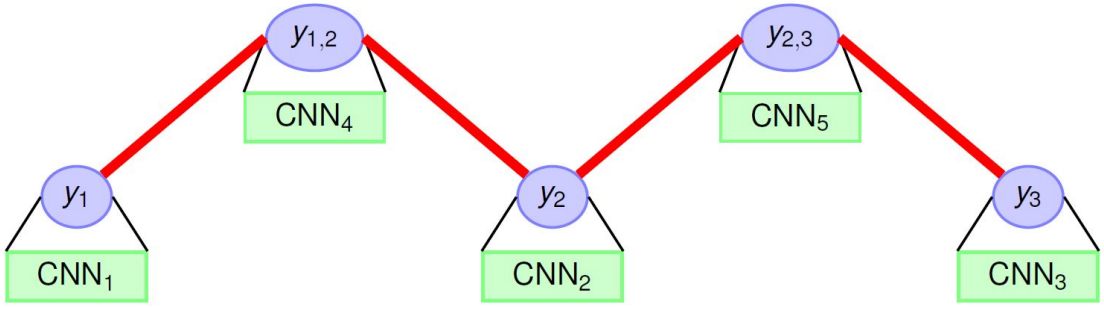

Figure 3: Each $f_r(\mathbf{w}, x, \mathbf{y}_r)$ represents a deep net

Now we are faced with the same problem as in the structured prediction: it's intractable to represent $F(\mathbf{w}, x^{(i)}, \hat{\mathbf{y}})$ as the dimension of output expands exponentially. One solution is scoring function decomposition as in (2): $F(\mathbf{w}, x, \mathbf{y}) = \sum_{r\in\mathcal{R}} f_r(\mathbf{w}, x, \mathbf{y}_r)$.
Here each element $f_r(\mathbf{w}, x, \mathbf{y}_r)$ represents an arbitrary composite function. It could represent a

deep network that takes in an image x and spit out segmentation labels for only two pixels. Given such decomposition, the gradient of cost function can be rewritten as:

$$\nabla_{\mathbf{w}} L(\mathbf{w}) = C\mathbf{w} + \sum_{i\in\mathcal{D}} \sum_{r\in\mathcal{R}} \sum_{\hat{\mathbf{y}}_r} \Big(p_r(\hat{\mathbf{y}}_r; x, \mathbf{w}) - \delta_r(\hat{\mathbf{y}}_r - \mathbf{y}_r^{(i)}) \Big) \nabla_{\mathbf{w}} f_r(\mathbf{w}, x^{(i)}, \hat{\mathbf{y}}_r) \tag{14}$$

where $p_r(\hat{\mathbf{y}}_r; x, \mathbf{w})$ represents the marginal probability:

$$p_r(\hat{\mathbf{y}}_r; x, \mathbf{w}) = \sum_{y_i \notin \mathbf{y}_r} p(\hat{\mathbf{y}}; x, \mathbf{w}) \tag{15}$$

5.2 Approximate local belief

The exact computation of this marginal probability is intractable (also due to the high dimension). To tackle this problem, [2] approximates true marginals $p_r(\hat{\mathbf{y}}_r; x, \mathbf{w})$ with local beliefs $b_r(\hat{\mathbf{y}}_r; x, \mathbf{w})$, and this local beliefs does not have to satisfy the above global constraint (15) but need to satisfy the following local constraint:

$$\mathcal{C}(x, \mathbf{y}) = \begin{cases} \forall r, & b_r(\hat{\mathbf{y}}_r; x, \mathbf{w}) \in \Delta \\ \forall r, \hat{\mathbf{y}}_r, p \in P(r) & \sum_{\hat{\mathbf{y}}_p \setminus \hat{\mathbf{y}}_r} b_p(\hat{\mathbf{y}}_p; x, \mathbf{w}) = b_r(\hat{\mathbf{y}}_r; x, \mathbf{w}) \end{cases} \tag{16}$$

where Δ is the probability simplex and $P(r)$ is the set of parents of region r, i.e.,$P(r) \subseteq \{p \in \mathcal{R}, r \subset p\}$, which subsumes those regions for which we want the marginalization constraint to hold. Now the learning problem can be rewritten as:

$$\min_{\mathbf{w}} \frac{C}{2}\|\mathbf{w}\|_2^2 + \sum_{i\in\mathcal{D}} \left(\max_{b_r(\hat{\mathbf{y}}_r; x, \mathbf{w}) \in \mathcal{C}(x,\mathbf{y})} \Big(\sum_{r\in\mathcal{R}} \sum_{\hat{\mathbf{y}}_r} b_r(\hat{\mathbf{y}}_r; x, \mathbf{w}) f_r(\mathbf{w}, x^{(i)}, \hat{\mathbf{y}}_r) \Big) - \sum_r f_r(\mathbf{w}, x^{(i)}, \mathbf{y}_r^{(i)}) \right) \tag{17}$$

For the sub maximization problem: (omitting $x, \mathbf{w}, \hat{\mathbf{y}} \to \mathbf{y}$ for simplicity)

$$\begin{aligned} &\max_{b_r} \sum_{r\in\mathcal{R}} \sum_{\mathbf{y}_r} b_r(\mathbf{y}_r) f_r(\mathbf{y}_r) \\ &\text{s.t.} \begin{cases} b_r(\mathbf{y}_r) \geq 0 & \forall r, \mathbf{y}_r \\ \sum_{\mathbf{y}_r} b_r(\mathbf{y}_r) = 1 & \forall r \\ \sum_{\mathbf{y}_p \setminus \mathbf{y}_r} b_p(\mathbf{y}_p) = b_r(\mathbf{y}_r) & \forall r, \mathbf{y}_r, p \in P(r) \end{cases} \end{aligned} \tag{18}$$

Using linear programming relaxation or message passing introduced in previous lectures, this approximated marginals could be used in the update step of weights.
The authors proved in [2] that using Lagrange dual, this minmax problem could be rewritten as a minimization problem with respect to $\mathbf{w}, \lambda$.

Theorem 1 *Assume $c_r \geq 0 \forall r$, Let $\lambda_{r\to p}(\mathbf{y}_r)$be the Lagrange multipliers for each marginalization constraint $\sum_{\mathbf{y}_p \setminus \mathbf{y}_r} b_p(\mathbf{y}_p) = b_r(\mathbf{y}_r)$ within the polytope $\mathcal{C}(x, \mathbf{y})$. Then the approximated general structured prediction task shown in Eq. (17) is equivalent to*

$$\min_{\mathbf{w},\lambda} \frac{C}{2}\|\mathbf{w}\|_2^2 + \sum_{i\in\mathcal{D}} \Big(\sum_r \ln \sum_{\mathbf{y}_r} \exp\big(\frac{\hat{f}_r(x^{(i)}, \mathbf{y}_r; \mathbf{w}, \lambda)}{c_r}\big) - F(\mathbf{w}, x^{(i)}, \mathbf{y}^{(i)}) \Big) \tag{19}$$

where re-parameterization score

$$\hat{f}_r(x^{(i)}, \mathbf{y}_r; \mathbf{w}, \lambda) = f_r(x^{(i)}, \mathbf{y}_r; \mathbf{w}) + \sum_{c\in C(r)} \lambda_{c\to r}(\mathbf{y}_c) - \sum_{p\in P(r)} \lambda_{r\to p}(\mathbf{y}_r) \tag{20}$$

5.3 Modified deep structured learning algorithm

Now each iteration in the optimization of decomposed scoring function operates as follow:

1. Forward passes to compute $f_r(\mathbf{w}, x^{(i)}, \hat{\mathbf{y}}_r)$
2. Estimate beliefs $b_r(\hat{\mathbf{y}}_r; x, \mathbf{w})$ exactly/approximately
3. Backward pass via chain-rule for gradient

$$\nabla_{\mathbf{w}} L(\mathbf{w}) = C\mathbf{w} + \sum_{i \in \mathcal{D}} \left(\sum_{r \in \mathcal{R}} \sum_{\hat{\mathbf{y}}_r} b_r(\hat{\mathbf{y}}_r; x, \mathbf{w}) \nabla_{\mathbf{w}} f_r(\mathbf{w}, x^{(i)}, \hat{\mathbf{y}}_r) - \nabla_{\mathbf{w}} F(\mathbf{w}, x^{(i)}, \mathbf{y}^{(i)}) \right) \quad (21)$$

4. Update parameters $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} L(\mathbf{w})$

6 Example: word prediction from distorted images

In [2], the authors experimented word recognition from noisy images. From fig.(4), we can see this task was very challenging, as these lower case character were taken from the original dataset and inserted to random background image patches by alpha matting, (characters has transparency), and images themselves were randomly scaled, rotated and translated.

Figure 4: Samples from the Word50 dataset. High degree of rotation, scaling and translation.

They experimented with graphical models composed of unary and pairwise regions defined over five random variables, one per character. The unary potentials $f_r(\mathbf{w}, x^{(i)}, \mathbf{y}_r^{(i)})$ was computed using multi-layer perceptrons (MLPs) with rectified linear units (ReLU). The pairwise potentials interactions via:

$$f_r(x, \mathbf{y}_i, \mathbf{y}_j; \mathbf{w}_p) = \sum_{m,n} W_{mn} \delta(\mathbf{y}_i = m, \mathbf{y}_j = n) \quad (22)$$

where $r = i, j, \mathbf{w}_p = W, W_{mn}$ is an element of W,and δ refers to the indicator function. For all experiments, all unary weights were shared across the nodes of the graphical model as well as all pairwise weights for all edges.
The first Markov model 5 has order one, i.e., there are links only between $\mathbf{y}_i$ and $\mathbf{y}_{i+1}$. The second model 6 has order 2, i.e., there are links only between $\mathbf{y}_i$ and $\mathbf{y}_{i+1}, \mathbf{y}_{i+2}$. They reported two metrics, the average character and word accuracy, which correspond to Hamming loss and zero-one loss respectively.

Their major findings are:

- **Joint training helps:** Joint training with pre-trained unary classifiers (pre-trains unaries but jointly optimizes pairwise weights as well as unary weights in a second step)outperforms all the other approaches in almost all cases. The piecewise training method (uses piecewise training by first training the unary potentials and then keeping them fixed when learning the pairwise potentials), unable to adapt the non-linearities while learning pairwise weights, often leads to performance worse than joint training.

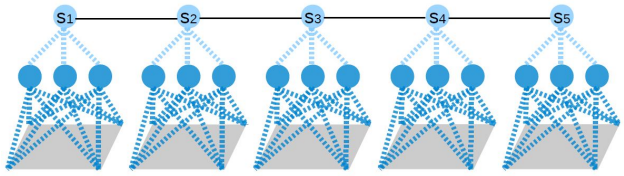

Figure 5: 1st order Markov graph model

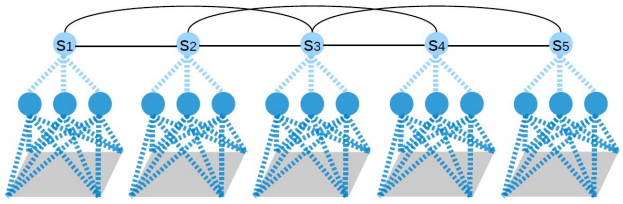

Figure 6: 2nd order Markov graph model

- **Structure helps:** Adding structure to the model is key to capture complex dependencies. As shown in 7, more structured models (i.e., second order Markov model and two-layer MLP unary potentials) consistently improves performance.

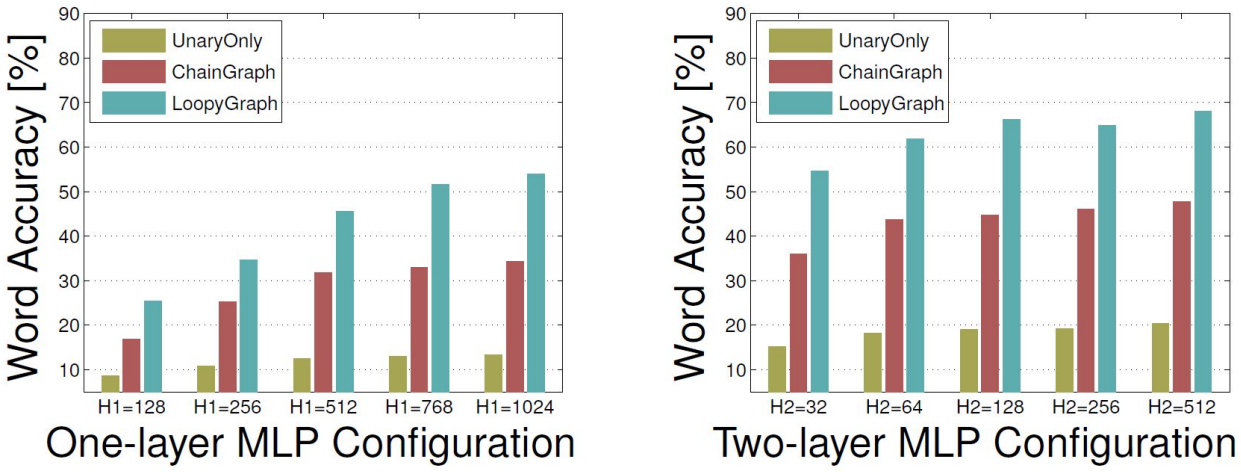

Figure 7: result

7 Example 2: Image segmentation

The author in [1] brings together methods from Deep Convolutional Neural Networks (DCNNs) and probabilistic graphical models for addressing the task of pixel-level classification. Combining the responses at the final DCNN layer with a fully connected Conditional Random Field (CRF) yield a better localization of segment boundaries.

As shown in Fig(8), a Deep Convolutional Neural Network such as VGG-16 or ResNet-101 is employed in a fully convolutional fashion, using atrous convolution to reduce the degree of signal downsampling (from 32x down 8x). A bilinear interpolation stage enlarges the feature maps to the original image resolution. A fully connected CRF is then applied to refine the segmentation result

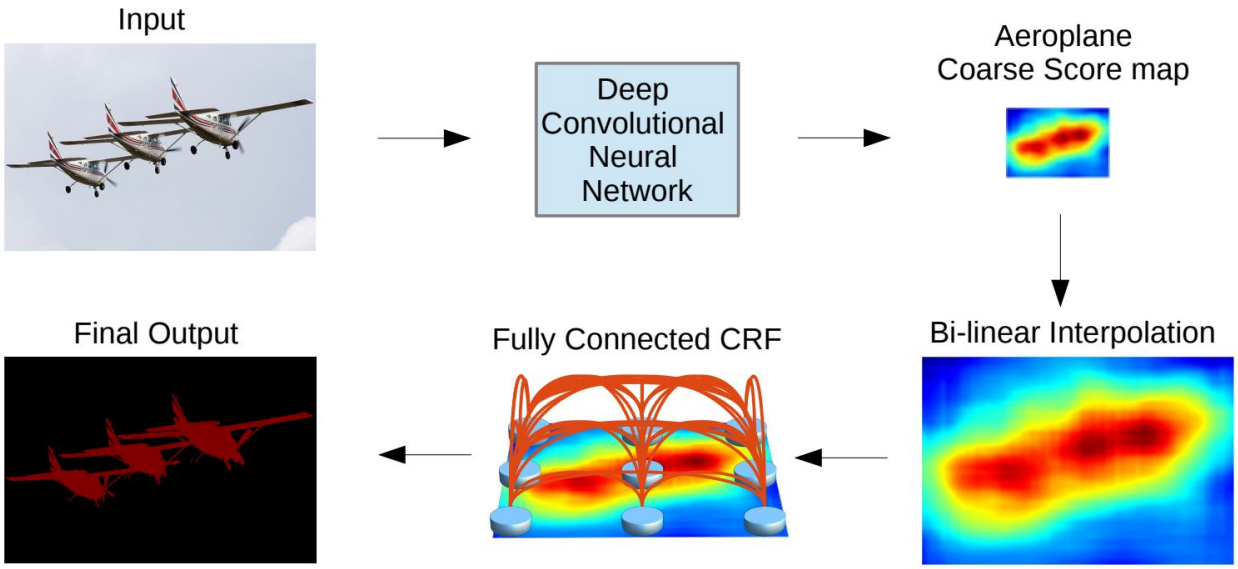

Figure 8: Model Illustration

and better capture the object boundaries.

Fig(9) shows how the score map (input before softmax function, first row) and belief map (output

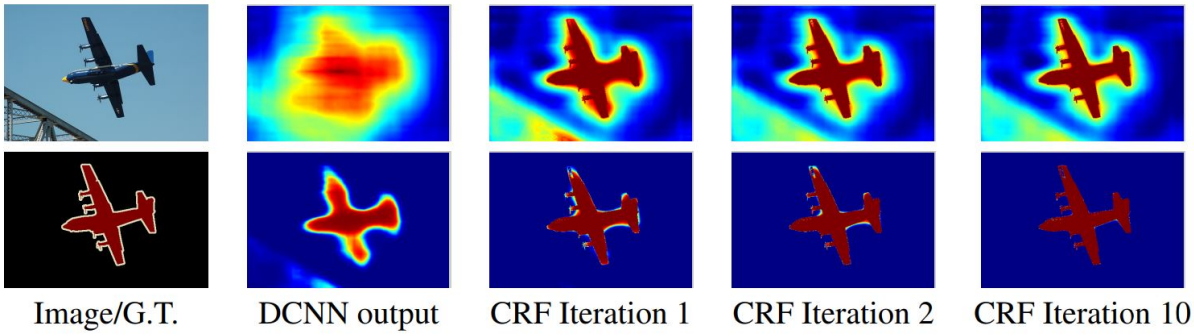

Figure 9: Score map and belief map

of softmax function, second) for Aeroplane changes along iterations. The output of last DCNN layer is used as input to the mean field inference.

8 Summary

This lectures deals with structured learning, where the output space grows exponentially. The main algorithm iterate between inference step eq.(23) and weight update step eq.(24). In the weight update step, local belief are iteratively estimated using Lagrange multiplier.

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} F(\mathbf{w}, x, \mathbf{y}) = \arg\max_{\mathbf{y}} \sum_{r \in \mathcal{R}} f_r(\mathbf{w}, x, \mathbf{y}_r) \tag{23}$$

$$\min_{\mathbf{w}} \frac{C}{2}\|\mathbf{w}\|_2^2 + \sum_{i\in\mathcal{D}} \epsilon \ln \sum_{\hat{y}} \exp \frac{L(\mathbf{y}^{(i)}, \hat{\mathbf{y}}) + F(\mathbf{w}, x^{(i)}, \hat{\mathbf{y}})}{\epsilon} - F(\mathbf{w}, x^{(i)}, \mathbf{y}^{(i)}) \tag{24}$$

The main idea behind this structured objects utilizes graphical model and decompose the scoring function into a linear combination of scoring function of model nodes.

References

[1] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[2] L.-C. Chen, A. Schwing, A. Yuille, and R. Urtasun. Learning deep structured models. In *International Conference on Machine Learning*, pages 1785–1794, 2015.