Optimization of CRF-as-RNN Hyperparameters using Bayesian Optimization

John Gibson
johngibson@wustl.edu

Abstract

In [1], Zheng et al. introduce a method for training Gaussian conditional random fields as recurrent neural network layers. Their method allows efficient training and inference of CRFs connected to standard classification networks, which improve semantic classification by enforcing smoothing constraints and pairwise compatibilities. Such CRFs are parametrized by appearance kernel parameters α and β, and smoothness parameters γ, as well as weightings for spatial and bilateral kernels used in the permutohedral lattice approximation for high-dimensional Gaussian features. The authors utilize hyperparameters derived from work in [2], on which their CRF formulation is based, but do not attempt to optimize these hyperparameters farther. This work presents a framework for Gaussian optimization of CRF hyperparameters to efficiently search a large parameter space for the parameter combination that optimizes the Jaccard (IoU) score of the resulting classifier. We implement this method and perform a parameter search, finding new CRF hyperparameters that result in a 2.7% increase in Jaccard score on a validation set from the Pascal VOC 2012 segmentation challenge.

1 Introduction

Semantic segmentation is a computationally challenging problem with many applications in computer vision. A semantic segmentation assigns to each pixel a label l from a set of possible labels $\mathcal{L}$, where each class corresponds to a set of discrete classifications for each pixel. Semantic segmentation finds uses in medical image processing, e.g. detection of malignant and benign tumors, defect detection in manufacturing, object detection in self-driving cars and other vehicles, and satellite imaging. Basic image segmentation used thresholding, contouring, and basic clustering methods, but these approaches were unsophisticated and error-prone. Early approaches using graph cuts and hand-coded descriptors performed better, but were quickly supplanted by neural-network based methods such as U-Net, SegNet, and DeepLab [3]. Deep learning approaches such as convolutional neural networks and fully connected networks produce per-pixel accuracies that are evaluated from local contexts, e.g. small image patches covered by a kernel. These methods perform well for many examples, but generally fail at image boundaries and do a poor job of utilizing global context and modelling pixel interactions. As such, many post-processing methods are used to smooth predictions, including bilateral filtering and other edge- and contour-based methods. A popular and effective method are Markov random fields and conditional random fields, which can encode relationships between pixels and thus construct consistent segmentations, but they are notoriously difficult to train and, crucially, are only trained on the output of the classification network itself. However, recent work modeling these processes as layers in neural networks has caused a resurgence in the use of MRFs and CRFs in image segmentation.

2 Background & Related Work

2.1 Conditional Random Fields

An MRF is a stochastic process defined on a graph G if and only if the probability of node x being assigned to value ω follows a Gibbs distribution over x and its clique $C(x)$, that is;

$$P(x=\omega) = \frac{1}{Z}\exp\left(-\sum_{x' \in C(x)} R(x', \omega)\right)$$

Where Z is a normalizing constant that forces the distribution over all ω to sum to 1, and $R(x', \omega)$ is the potential value of clique member x' if x is assigned to label ω [**?**]. A useful property is to model can be split into *unary* and *pairwise* potentials:

$$P(x=\omega) = \frac{1}{Z}\exp\left(-U(x,\omega) - \sum_{x' \neq x \in C(x)} V(x, x', \omega)\right)$$

Where $U(x,\omega)$ is the unary potential of assigning label ω to x, and $V(x, x', \omega)$ is the pairwise potential of assigning ω to x given the values of each other x' in x's clique.

Many methods exist to optimize these probabilistic structures over label assignments.

Conditional random fields (CRFs) are a special type of MRF whose potentials are conditioned on input data X and directly model $P(x = \omega|X)$:

$$P(x = \omega|X) = \frac{1}{Z} \prod_{x' \in C(x)} \psi(x = \omega|x', X) \tag{1}$$

Where $\psi(x = \omega|x', X)$ is the potential of assigning x to ω given x' and input values X. Similarly, this can be split into unary and pairwise potential terms.

2.2 Use of Conditional Random Fields in Neural Networks

As such, these methods dropped in popularity as pure neural-network approaches became better at semantic segmentation. However, in 2015, Zheng *et al.* presented a formulation of a fully-connected Gaussian CRF for image segmentation that is modeled as layers of a recurrent neural network (RNN) [1]. As such, the CRF layer(s) can be placed at the end of a standard segmentation network and trained end-to-end, thus allowing efficient training of the CRF alongside the neural network and allowing the neural network itself to take advantage of the smoothing of the CRF. Zheng *et al.* adapt a CRF formulation presented by Krahenbuhl and Koltun [2], which receives the unary potentials from an external classifier (in the adapted case, a neural network) and defines the pairwise potentials as a linear combination of an appearance kernel and a smoothing kernel, which are Gaussian and parametrized by $\alpha \& \beta$ and γ, respectively:

$$\begin{aligned}
\psi_p(\mathbf{x}_i, \mathbf{x}_j) &= c(x_i, x_j) k(\mathbf{f}_i, \mathbf{f}_j) \\
k(\mathbf{f}_i, \mathbf{f}_j) &= \underbrace{w^{(1)} \exp\left(-\frac{|p_i - p_j|^2}{2\alpha^2} - \frac{|I_i - I_j|^2}{2\beta^2}\right)}_{\text{appearance}} \\
&\quad + \underbrace{w^{(2)} \exp\left(-\frac{|p_i - p_j|^2}{2\gamma^2}\right)}_{\text{smoothing}}
\end{aligned} \tag{2}$$

Where $\mathbf{f}_i$ is an augmented vector containing the positions of the pixel p_i and the intensity values of the pixel I_i, and the hypothetical label x_i. $c(x_i, x_j)$ is a compatibility penalty that regulates which pixel classes are likely to be found next to each other (e.g. car and road would have low penalty, whereas car and airplane would have a higher penalty). The appearance kernel enforces that regions close together have similar appearances, and the smoothing kernel helps eliminate small regions.

$\alpha, \beta,$ and γ are all kernel width parameters,and $w^{(1)}, w^{(2)}$ are weighting parameters. Zheng *et al.* set default parameters of $\alpha = 160, \beta = 3, \gamma = 3, w^{(1)} = 5, w^{(2)} = 3$. Krahenbuhl and Koltun set $\alpha = 61, \beta = 11, \gamma = 1, w^{(1)} = 1, w^{(2)} = 1$.

Once trained, inference proceeds through a mean-field approximation, which minimizes the KL-divergence between the exact distribution $P(X)$ and a product of marginals $Q(X) = \prod_i Q_i(X_i)$ via an iterative process of message passing, compatibility transformation, and local update. This process can be performed as a stack of RNN layers, as seen in algorithm 1.

Algorithm 1 Mean-field inference as RNN layers, where l is a label, $U_i(l)$ is the unary potential for pixel i for label l, and Z_i is the normalizing constant for pixel i.

```
$Q_i(l) \leftarrow \frac{1}{Z_i} \exp(U_i(l))$ for all $i$
while not converged do
    $A_i^{(m)} \leftarrow \sum_{j \neq i} k^{(m)}(\mathbf{f}_i, \mathbf{f}_j) Q_j(l)$ for all $m$
        ▷ message passing
    $B_i(l) \leftarrow \sum_m w^{(m)} A_i^{(m)}(l)$
        ▷ weighting filter outputs
    $C_i(l) \leftarrow \sum_{l' \in \mathcal{L}} c(l, l') B_i(l')$
        ▷ compatibility transform
    $D_i(l) \leftarrow U_i(l) - C_i(l)$
        ▷ add unary potentials
    $Q_i \leftarrow \frac{1}{Z_i} \exp(D_i(l))$
        ▷ normalize via softmax
end while
```

For additional efficiency, Gaussian filtering is performed using the permutohedral lattice approximation, which allows for $O(N)$ filtering time and backpropagation. The authors note that this implementation generally converges in < 10 iterations.

2.3 Bayesian Optimization

Bayesian optimization is a noise-robust framework for optimizing functions of low- to mid-dimension with expensive evaluations over continuous domains [4]. The process builds a surrogate model of the function value across the domain, calculates the uncertainty at each point, and determines the next point to sample by maximizing an aquisition function over the surrogate model. A general description of the process can be found in algorithm 2. A Gaussian process, which is relatively easy to train, produces results robust to observation noise, provides a method of estimating uncertainty, and can model complex functions with the correct kernel, is an excellent model for this task. The core algorithm works as follows:

1. The Gaussian process is seeded with initial data and trained.

2. An acquisition function, which scores each point on the domain in terms of its likelihood to improve the

metric score, is maximized over the domain. The acquisition function only acts on the mean, μ, and variance, σ^2, of the GP at each point, and thus is relatively fast to optimize.

3. The point which maximizes the acquisition function is chosen, and the true (expensive) function is evaluated at that point.

4. The new point is added to the dataset, and the GP is retrained.

5. Repeat for N iterations or until some threshold is met.

This process is limited to around 20 parameters, as the Gaussian process is not necessarily convex, and the optimization process would generally find only local minima due to the curse of dimensionality. However, for hyperparameter training, which generally has few parameters, this process works exceptoinally well.

Algorithm 2 Generalized Bayesian optimization framework

```
Initialize $\mathbf{X}$ with $N_i$ randomly chosen points from the domain $D$.
Place a Gaussian process prior on $f(\mathbf{X})$.
for $i \in 1..N$ do
    $x_i \leftarrow \arg\max_x A(x)$    $\triangleright$ maximize acquisition fn
    $y_i \leftarrow f(x_i)$
    $\mathbf{X} \leftarrow \mathbf{X} \cup (x_i, y_i)$
    Update GP over $\mathbf{X}$
end for
return maximum value of $y$ in $\mathbf{X}$
```

A general-purpose acquisition function, and the one used in this work, is expected improvement. We define this metric as:

$$\mathbf{EI}(x) = \mathbb{E}\left[\max(g(x) - \mathbf{f}^*, 0)\right] \tag{3}$$

Where $g(x)$ is the GP estimate at x, and $\mathbf{f}^*$ is the maximum value from all evaluations of $f(\mathbf{X})$. As $g(x)$ is normally-distributed with mean $\mu(x)$ and variance $\sigma^2(x)$, we can solve for $\mathbf{EI}(x)$ in closed-form:

$$\begin{aligned}\Delta(x) &= \mu(x) - \mathbf{f}^* \\ \mathbf{EI}(x) &= \max(\Delta(x), 0) + \sigma(x)\phi\left(\frac{\Delta(x)}{\sigma(x)}\right) \\ &\quad - |\Delta(x)|\Phi\left(\frac{\Delta(x)}{\sigma(x)}\right)\end{aligned} \tag{4}$$

We then choose the x that maximizes the expected improvement metric and sample the true value of f at that location. Finally, we simply return the best value of $f(x)$ that we have found throughout the entire process. In this case, the metric we optimize over is the Jaccard index:

$$\text{Jaccard}(A, B) = \frac{\sum_i I[A_i = B_i]}{\sum_i I[A_i \neq 0 || B_i \neq 0]} \tag{5}$$

where $I[q]$ is the indicator bracket.

The Jaccard index, also known as the intersection over union (IoU) metric, measures the ratio of true positive to the number of pixels that are labelled in either the output label image or the ground truth. The metric scales from 0 to 1, with higher values indicating better match to the ground truth.

3 Methods

3.1 Network

Our implementation is based on the network of Zheng *et al.*, with minor modifications. Their network uses a fully-connected network called FCN-8S and directly concatenates their novel CRF-as-RNN layers onto its outputs. The network was retrained to take full advantage of the CRF smoothing, and achieved an average Jaccard index of 0.696 on the Pascal VOC 2012 validation dataset. We similarly use FCN-8S as the base neural network classifier, and use the trained weights provided by Zheng *et al.* for the full network. The CRF-as-RNN layers were modified to allow for swapping of parameters without rebuilding the model, and the permutohedral implementation was slightly modified to fix a compilation error. The number of iterations in the mean-field layer was set to 10 for all tests. To match the Pascal VOC training dataset, the number of classes was set to 21. All other parameters, including label compatibilities, are identical to those in Zheng *et al.*.

3.2 Bayesian Optimization

We utilize scikit-learn's implementation of Gaussian Processes for this work. Specifically, we implement a general framework for Bayesian optimization as follows:

- By default, the Matern kernel is used with $\nu = 1.5$, and the default L-BFGS optimizer is used to optimize the kernel hyperparameters with n restarts (default 10), with `normalize_y` set to `True`.

- α is fixed for all trials (default 1e-5).

- N_i initialization samples (default 10) are randomly sampled from the domain, and the process is run for N iterations (default 50).

- The domain is bounded (default between 0 and 1 for all parameters), and a separate scale parameter for each dimension is used to scale the value of x returned by the optimization before the true value of $f(x)$ is evaluated. This normalization helps the numerical stability and accuracy of the GP.

- Expected improvement (equation 3) is used as the acquisition function.
- Hyperparameters were chosen by maximization of $g(x)$ over the entire domain by L-BFGS-B with N_r random restarts (default 100), and the optimal point from all restarts is returned.
- Duplicates in the GP are avoided by sampling randomly from the domain if the selected point is within some small ϵ (default 1e-7).

3.3 Training and Metrics

Average IoU (Jaccard index) is used as the objective function for Bayesian optimization by simply averaging Jaccard indices (equation 4) over a validation set ($n = 10$, in this case, but with ample computational resources at support for batching this could theoretically scale up to the entire training dataset). For this implementation, scales were set at 500 for α, and 10 for all other parameters. This is consistent with the larger values for α chosen by Zheng *et al.*.

3.4 Validation

After training, average IoU is calcluated for a holdout validation set and reported, and the new parameters are saved. In addition, for evaluation, the trained optimal parameters are varied and average IoU plotted for values along each dimension. This does not capture interactions between parameters, but can give a rough estimation of how the metric varies for each dimension as well as the value of the optimized parameters.

4 Experimental Results

A set of 10 training images and 10 validation images was selected from the Pascal VOC 2012 dataset, and Bayesian optimization was run on the training data as described in the Methods section with default parameters. Training took approximately 4 hours on a consumer-grade laptop. After 50 iterations, performance on the validation set had risen from 0.736 average IoU to 0.762 average IoU, an accuracy gain of 2.7%. The parameter changes are summarized in table 1.

Images processed with the default parameters and the optimized parameters are compared with the ground truth in figure 2. Figure 1 shows the accuracy results on the validation set with varying parameters, as discussed above.

Parameter	**Default**	**Optimized**
α	160.0	96.5
β	3.0	9.36
γ	3.0	9.37
$w^{(1)}$	5.0	3.51
$w^{(2)}$	3.0	0.02
Average IoU	0.73619	0.76377

Table 1: Optimized parameters resulted in a 2.7% increase in average IoU. Note that the weighting of the spatial kernel is close to zero; it is possible that the first term inside the appearance kernel is sufficient for enforcing smoothness.

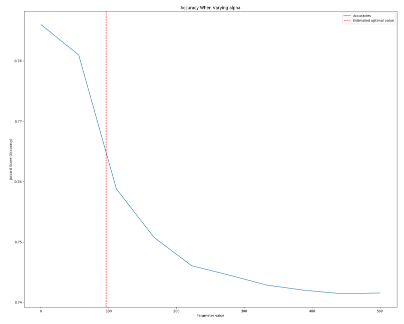
5 Further Work

In the future, we hope to scale up to be able to optimize these parameters over the entire training dataset. As correspondingly fewer iterations are needed as compared to a grid search, this is actually computationally feasible. In addition, other acquisition functions could be tried, such as knowledge gradient, and the number of iterations could be increased. Different kernels, scales, and boundaries could also be tested; other metrics, such as Dice score, could also be tested. The response of the metric to changes in parameters is currently run independently for each parameter; a better sense of how these parameters are related could be deduced through pairwise comparisons (perhaps also through Bayesian methods! e.g. Bayesian quadrature). Finally, the updated parameters could be evaluated on the entire validation dataset to test the true performance increase.

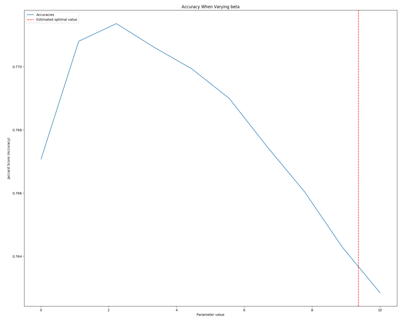
The network architecture could also be improved; prediction is currently slow due to the lack of batching support by the CRF-as-RNN layers. This could fairly easily be improved, decreasing both training and inference time.

6 Conclusion

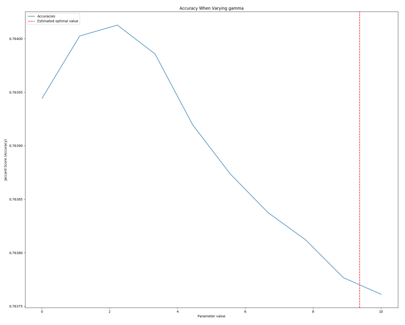
In this work, we utilize Bayesian optimization to tune the hyperparameters of a conditional random field and demonstrate its efficacy on a small validation dataset. We also develop a general framework for Bayesian optimization of hyperparameters of image-processing algorithms, using the expected improvement function and the Jaccard index. We compare the results of the default parameters and the optimized parameters and find that the optimized parameters seem to produce edges that more closely match those of the true objects and have fewer small areas with spurious labels.



(a) α



(b) β



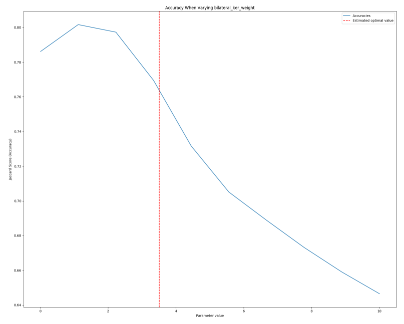
(c) γ

Figure 1: Changes in average IoU of the validation set as each parameter is varied. The value chosen by Bayesian optimization on the training set is maked by a dashed red line.

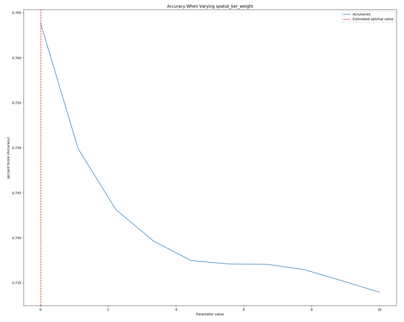
Acknowledgments

Thanks to Mark Hua for computational resources.

Thanks to Thomas Huijskens for his article on Bayesian optimization.



(d) $w^{(1)}$



(e) $w^{(2)}$

Figure 1: Changes in average IoU of the validation set as each parameter is varied (cont.)

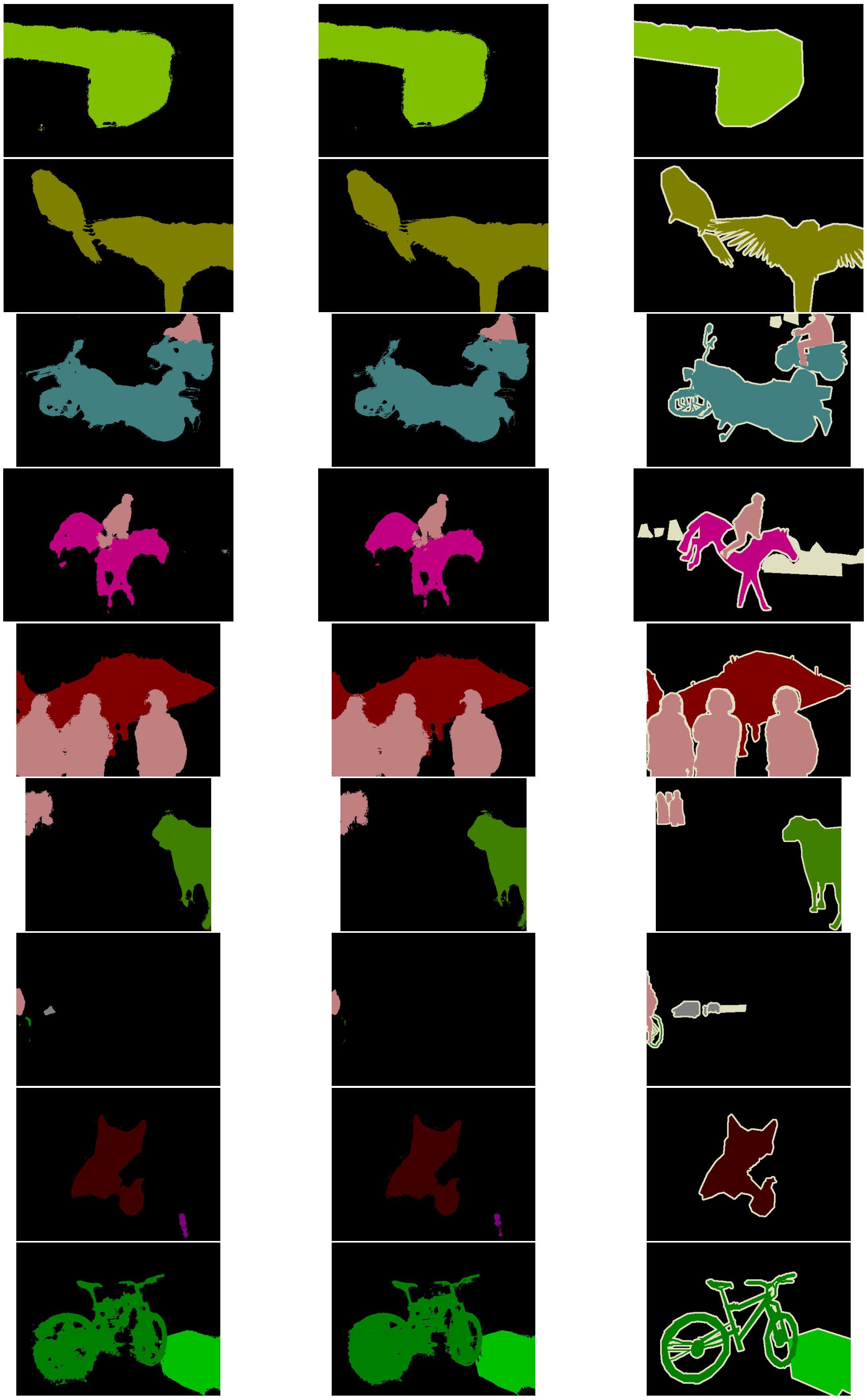
References

[1] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. Conditional random fields as recurrent neural networks. *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015.

[2] Philipp Krhenbhl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials, 2012.

[3] Irem Ulku and Erdem Akagunduz. A survey on deep learning-based architectures for semantic segmentation on 2d images, 2019.

[4] Peter I. Frazier. A tutorial on bayesian optimization, 2018.



(a) Default parameters (b) Optimized parameters (c) Ground truth

Figure 2: Comparison of results from default parameters, trained parameters, and ground truth on a validation set. The optimized parameters seem to enforce smoother image boundaries with fewer small regions.