Fast, Exact and Multi-Scale Inference for Semantic Image Segmentation with Deep Gaussian CRFs

Siddhartha Chandra
siddhartha.chandra@inria.fr

Iasonas Kokkinos
iasonas.kokkinos@ecp.fr

INRIA GALEN & Centrale Supélec, Paris, France

Abstract. In this work we propose a structured prediction technique that combines the virtues of Gaussian Conditional Random Fields (G-CRF) with Deep Learning: (a) our structured prediction task has a unique global optimum that is obtained exactly from the solution of a linear system (b) the gradients of our model parameters are analytically computed using closed form expressions, in contrast to the memory-demanding contemporary deep structured prediction approaches [1,2] that rely on back-propagation-through-time, (c) our pairwise terms do not have to be simple hand-crafted expressions, as in the line of works building on the DenseCRF [1,3], but can rather be 'discovered' from data through deep architectures, and (d) out system can trained in an end-to-end manner. Building on standard tools from numerical analysis we develop very efficient algorithms for inference and learning, as well as a customized technique adapted to the semantic segmentation task. This efficiency allows us to explore more sophisticated architectures for structured prediction in deep learning: we introduce multi-resolution architectures to couple information across scales in a joint optimization framework, yielding systematic improvements. We demonstrate the utility of our approach on the challenging VOC PASCAL 2012 image segmentation benchmark, showing substantial improvements over strong baselines. We make all of our code and experiments available at https://github.com/siddharthachandra/gcrf.

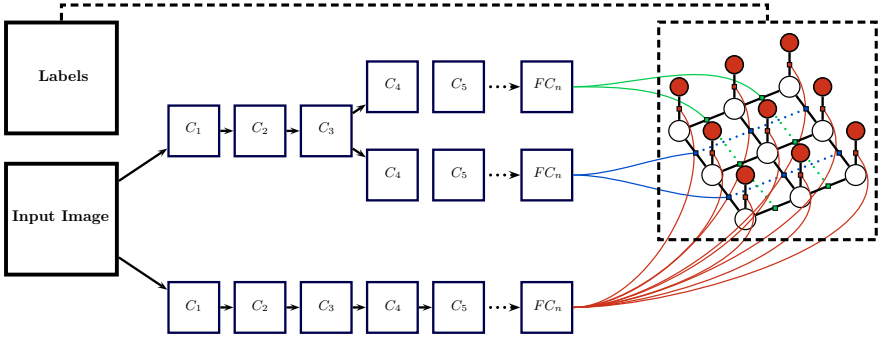
1 Introduction

Over the last few years deep learning has resulted in dramatic progress in the task of semantic image segmentation. Early works on using CNNs as feature extractors [4,5,6] and combining them with standard superpixel-based front-ends gave substantial improvements over well-engineered approaches that used hand-crafted features. The currently mainstream approach is relying on 'Fully' Convolutional Networks (FCNs) [7,8], where CNNs are trained to provide fields of outputs used for pixelwise labeling.

A dominant research direction for improving semantic segmentation with deep learning is the combination of the powerful classification capabilities of FCNs with structured prediction [1,2,3,9,10,11], which aims at improving classification by capturing interactions between predicted labels. One of the first works in the direction of combining deep networks with structured prediction was [3] which advocated the use of densely-connected conditional random fields (DenseCRF) [12] to post-process an FCNN output so as to obtain a sharper segmentation the preserves image boundaries. This was then

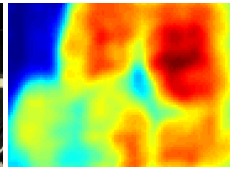
used by Zheng *et al*. [1] who combined DenseCRF with a CNN into a single Recurrent Neural Network (RNN), accommodating the DenseCRF post processing in an end-to-end training procedure.

Most approaches for semantic segmentation perform structured prediction using approximate inference and learning [9,13]. For instance the techniques of [1,2,3,10] perform mean-field inference for a fixed number of 10 iterations. Going for higher accuracy with more iterations could mean longer computation and eventually also memory bottlenecks: back-propagation-through-time operates on the intermediate 'unrolled inference' results that have to be stored in (limited) GPU memory. Furthermore, the non-convexity of the mean field objective means more iterations would only guarantee convergence to a local minimum. The authors in [14] use piecewise training with

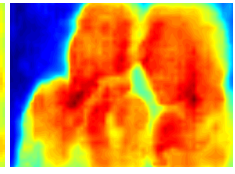


1(a) Schematic of a fully convolutional neural network with a G-CRF module

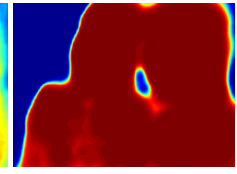
1(b) Input Image



1(c) Person unary



1(d) QO Output



1(e) Person Probability

Fig. 1: (a) shows a detailed schematic representation of our fully convolutional neural network with a G-CRF module. The G-CRF module is shown as the box outlined by dotted lines. The factor graph inside the G-CRF module shows a $4-$connected neighbourhood. The white blobs represent pixels, red blobs represent unary factors, the green and blue squares represent vertical and horizontal connectivity factors. The input image is shown in (b). The network populates the unary terms (c), and horizontal and vertical pairwise terms. The G-CRF module collects the unary and pairwise terms from the network and proposes an image hypothesis, i.e. scores (d) after inference. These scores are finally converted to probabilities using the Softmax function (e), which are then thresholded to obtain the segmentation. It can be seen that while the unary scores in (c) miss part of the torso because it is occluded behind the hand. The flow of information from the neighbouring region in the image, via the pairwise terms, encourages pixels in the occluded region to take the same label as the rest of the torso (d). Further it can be seen that the person boundaries are more pronounced in the output (d) due to pairwise constraints between pixels corresponding to the person and background classes.

CNN-based pairwise potentials and three iterations of inference, while those in [15] use highly-sophisticated modules, effectively learning to approximate mean-field inference. In these two works a more pragmatic approach to inference is taken, considering it as a sequence of operations that need to be learned [1]. These 'inferning'-based approaches of combining learning and inference may be liberating, in the sense that one acknowledges and accommodates the approximations in the inference through end-to-end training. We show however here that exact inference and learning is feasible, while not making compromises in the model's expressive power.

Motivated by [16,17], our starting point in this work is the observation that a particular type of graphical model, the Gaussian Conditional Random Field (G-CRF), allows us to perform exact and efficient Maximum-A-Posteriori (MAP) inference. Even though Gaussian Random Fields are unimodal and as such less expressive, Gaussian *Conditional* Random Fields are unimodal *conditioned on the data*, effectively reflecting the fact that given the image one solution dominates the posterior distribution. The G-CRF model thus allows us to construct rich expressive structured prediction models that still lend themselves to efficient inference. In particular, the log-likelihood of the G-CRF posterior has the form of a quadratic energy function which captures unary and pairwise interactions between random variables. There are two advantages to using a quadratic function: (a) unlike the energy of general graphical models, a quadratic function has a unique global minimum if the system matrix is positive definite, and (b) this unique minimum can be efficiently found by solving a system of linear equations. We can actually discard the probabilistic underpinning of the G-CRF and understand G-CRF inference as an energy-based model, casting structured prediction as quadratic optimization (QO).

G-CRFs were exploited for instance in the regression tree fields model of Jancsary *et al*. [17] where decision trees were used to construct G-CRF's and address a host of vision tasks, including inpainting, segmentation and pose estimation. In independent work [2] proposed a similar approach for the task of image segmentation with CNNs, where as in [14,15,18] FCNs are augmented with discriminatively trained convolutional layers that model and enforce pairwise consistencies between neighbouring regions.

One major difference to [2], as well as other prior works [1,3,10,14,15], is that we use exact inference and do not use back-propagation-through-time during training. In particular building on the insights of [16,17], we observe that the MAP solution, as well as the gradient of our objective with respect to the inputs of our structured prediction module can be obtained through the solution of linear systems. Casting the learning and inference tasks in terms of linear systems allows us to exploit the wealth of tools from numerical analysis. As we show in Sec. 3, for Gaussian CRFs sequential/parallel mean-field inference amounts to solving a linear system using the classic Gauss-Seidel/Jacobi algorithms respectively. Instead of these under-performing methods we use conjugate gradients which allow us to perform exact inference and back-propagation in a small number (typically 10) iterations, with a negligible cost (0.02s for the general case in Sec. 2, and 0.003s for the simplified formulation in Sec. 2.5) when implemented on the GPU.

Secondly, building further on the connection between MAP inference and linear system solutions, we propose memory- and time-efficient algorithms for weight-sharing

(Sec. 2.5) and multi-scale inference (Sec. 3.2). In particular, in Section 2.5 we show that one can further reduce the memory footprint and computation demands of our method by introducing a Potts-type structure in the pairwise term. This results in multifold accelerations, while delivering results that are competitive to the ones obtained with the unconstrained pairwise term. In Sec. 3.2 we show that our approach allows us to work with arbitrary neighbourhoods that go beyond the common $4-$connected neighbourhoods. In particular we explore the merit of using multi-scale networks, where variables computed from different image scales interact with each other. This gives rise to a flow of information across different-sized neighborhoods. We show experimentally that this yields substantially improved results over single-scale baselines.

In Sec. 2 we describe our approach in detail, and derive the expressions for weight update rules for parameter learning that are used to train our networks in an end-to-end manner. In Sec. 3 we analyze the efficiency of the linear system solvers and present our multi-resolution structured prediction algorithm. In Sec. 4 we report consistent improvements over well-known baselines and state-of-the-art results on the VOC PASCAL test set.

2 Quadratic Optimization Formulation

We now describe our approach. Consider an image $\mathcal{I}$ containing P pixels. Each pixel $p \in \{p_1, \ldots, p_P\}$ can take a label $l \in \{1, \ldots, L\}$. Although our objective is to assign discrete labels to the pixels, we phrase our problem as a continuous inference task. Rather than performing a discrete inference task that delivers one label per variable, we use a continuous function of the form $\mathbf{x}(p, l)$ which gives a score for each pairing of a pixel to a label. This score can be intuitively understood as being proportional to the log-odds for the pixel p taking the label l, if a 'softmax' unit is used to post-process $\mathbf{x}$.

We denote the pixel-level ground-truth labeling by a discrete valued vector $\mathbf{y} \in \mathbb{Y}^P$ where $\mathbb{Y} \in \{1, \ldots, L\}$, and the inferred hypothesis by a real valued vector $\mathbf{x} \in \mathbb{R}^N$, where $N = P \times L$. Our formulation is posed as an energy minimization problem. In the following subsections, we describe the form of the energy function, the inference procedure, and the parameter learning approach, followed by some technical details pertinent to using our framework in a fully convolutional neural network. Finally, we describe a simpler formulation with pairwise weight sharing which achieves competitive performance while being substantially faster. Even though our inspiration was from the probabilistic approach to structured prediction (G-CRF), from now on we treat our structured prediction technique as a Quadratic Optimization (QO) module, and will refer to it as QO henceforth.

2.1 Energy of a hypothesis

We define the energy of a hypothesis in terms of a function of the following form:

$$E(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T(A + \lambda\mathbf{I})\mathbf{x} - B\mathbf{x} \tag{1}$$

where A denotes the symmetric $N \times N$ matrix of pairwise terms, and B denotes the $N \times 1$ vector of unary terms. In our case, as shown in Fig. 1, the pairwise terms A and

the unary terms B are learned from the data using a fully convolutional network. In particular and as illustrated in Fig. 1, A and B are the outputs of the pairwise and unary streams of our network, computed by a forward pass on the input image. These unary and pairwise terms are then combined by the QO module to give the final per-class scores for each pixel in the image. As we show below, during training we can easily obtain the gradients of the output with respect to the A and B terms, allowing us to train the whole network end-to-end.

Eq. 1 is a standard way of expressing the energy of a system with unary and pairwise interactions among the random variables [17] in a vector labeling task. We chose this function primarily because it has a unique global minimum and allows for exact inference, alleviating the need for approximate inference. Note that in order to make the matrix A strictly positive definite, we add to it λ times the Identity Matrix $\mathbf{I}$, where λ is a design parameter set empirically in the experiments.

2.2 Inference

Given A and B, inference involves solving for the value of $\mathbf{x}$ that minimizes the energy function in Eq. 1. If $(A + \lambda\mathbf{I})$ is symmetric positive definite, then $E(\mathbf{x})$ has a unique global minimum [19] at:

$$(A + \lambda\mathbf{I})\mathbf{x} = B. \tag{2}$$

As such, inference is exact and efficient, only involving a system of linear equations.

2.3 Learning A and B

Our model parameters A and B are learned in an end-to-end fashion via the back-propagation method. In the back-propagation training paradigm each module or *layer* in the network receives the derivative of the final loss $\mathcal{L}$ with respect to its output $\mathbf{x}$, denoted by $\frac{\partial\mathcal{L}}{\partial\mathbf{x}}$, from the layer above. $\frac{\partial\mathcal{L}}{\partial\mathbf{x}}$ is also referred to as the gradient of $\mathbf{x}$. The module then computes the gradients of its inputs and propagates them down through the network to the layer below.

To learn the parameters A and B via back-propagation, we require the expressions of gradients of A and B, i.e. $\frac{\partial\mathcal{L}}{\partial A}$ and $\frac{\partial\mathcal{L}}{\partial B}$ respectively. We now derive these expressions.

Derivative of Loss with respect to B To compute the derivative of the loss with respect to B, we use the chain rule of differentiation: $\frac{\partial\mathcal{L}}{\partial\mathbf{x}} = \frac{\partial\mathcal{L}}{\partial B}\frac{\partial B}{\partial\mathbf{x}}$. Application of the chain rule yields the following closed form expression, which is a system of linear equations:

$$(A + \lambda\mathbf{I})\frac{\partial\mathcal{L}}{\partial B} = \frac{\partial\mathcal{L}}{\partial\mathbf{x}}. \tag{3}$$

When training a deep network, the right hand side $\frac{\partial\mathcal{L}}{\partial B}$ is delivered by the layer above, and the derivative on the left hand side is sent to the unary layer below.

Derivative of Loss with respect to A The expression for the gradient of A is derived by using the chain rule of differentiation again: $\frac{\partial \mathcal{L}}{\partial A} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial A}$.

Using the expression $\frac{\partial \mathbf{x}}{\partial A} = \frac{\partial}{\partial A}(A+\lambda \mathbf{I})^{-1} B$, substituting $\frac{\partial}{\partial A}(A+\lambda \mathbf{I})^{-1} = -(A+\lambda \mathbf{I})^{-T} \otimes (A+\lambda \mathbf{I})^{-1}$, and simplifying the right hand side, we arrive at the following expression:

$$\frac{\partial \mathcal{L}}{\partial A} = -\frac{\partial \mathcal{L}}{\partial B} \otimes \mathbf{x}, \tag{4}$$

where $\otimes$ denotes the kronecker product. Thus, the gradient of A is given by the negative of the kronecker product of the output $\mathbf{x}$ and the gradient of B.

2.4 Softmax Cross-Entropy Loss

Please note that while in this work we use the QO module as the penultimate layer of the network, followed by the softmax cross-entropy loss, it can be used at any stage in a network and not only as the final classifier. We now give the expressions for the softmax cross-entropy loss and its derivative for sake of completeness.

The image hypothesis is a scoring function of the form $\mathbf{x}(p, l)$. For brevity, we denote the hypothesis concerning a single pixel by $\mathbf{x}(l)$. The softmax probabilities for the labels are then given by $p_l = \frac{e^{\mathbf{x}(l)}}{\sum_L e^{\mathbf{x}(l)}}$. These probabilities are penalized by the cross-entropy loss defined as $\mathcal{L} = -\sum_l \mathbf{y}_l \log p_l$, where $\mathbf{y}_l$ is the ground truth indicator function for the ground truth label l^*, i.e. $\mathbf{y}_l = 0$ if $l \neq l^*$, and $\mathbf{y}_l = 1$ otherwise. Finally the derivative of the softmax-loss with respect to the input is given by: $\frac{\partial \mathcal{L}}{\partial \mathbf{x}(l)} = p_l - y_l$.

2.5 Quadratic Optimization with Shared Pairwise Terms

We now describe a simplified *QO* formulation with shared pairwise terms which is significantly faster in practice than the one described above. We denote by $A_{p_i,p_j}(l_i, l_j)$ the pairwise energy term for pixel p_i taking the label l_i, and pixel p_j taking the label l_j. In this section, we propose a *Potts*-type pairwise model, described by the following equation:

$$A_{p_i,p_j}(l_i, l_j) = \left\{ \begin{array}{ll} 0 & l_i = l_j \\ A_{p_i,p_j} & l_i \neq l_j. \end{array} \right\} \tag{5}$$

In simpler terms, unlike in the general setting, the pairwise terms here depend on whether the pixels take the same label or not, and not on the particular labels they take. Thus, the pairwise terms are *shared* by different pairs of classes. While in the general setting we learn $PL \times PL$ pairwise terms, here we learn only $P \times P$ terms. To derive the inference and gradient equations after this simplification, we rewrite our inference equation $(A + \lambda \mathbf{I})\,\mathbf{x} = B$ as,

$$\begin{bmatrix} \lambda \mathbf{I} & \hat{A} & \cdots & \hat{A} \\ \hat{A} & \lambda \mathbf{I} & \cdots & \hat{A} \\ & & \vdots & \\ \hat{A} & \hat{A} & \cdots & \lambda \mathbf{I} \end{bmatrix} \times \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_L \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_L \end{bmatrix} \tag{6}$$

where $\mathbf{x}_k$, denotes the vector of scores for all the pixels for the class $k \in \{1, \cdots, L\}$. The per-class unaries are denoted by $\mathbf{b}_k$, and the pairwise terms $\hat{A}$ are shared between each pair of classes. The equations that follow are derived by specializing the general inference (Eq. 2) and gradient equations (Eq. 3,4) to this particular setting. Following simple manipulations, the inference procedure becomes a two step process where we first compute the sum of our scores $\sum_i \mathbf{x}_i$, followed by $\mathbf{x}_k$, i.e. the scores for the class k as:

$$\left(\lambda \mathbf{I} + (L-1)\,\hat{A}\right) \sum_i \mathbf{x}_i = \sum_i \mathbf{b}_i, \quad (7) \qquad (\lambda \mathbf{I} - \hat{A})\mathbf{x}_k = \mathbf{b}_k - \hat{A} \sum_i \mathbf{x}_i. \quad (8)$$

Derivatives of the unary terms with respect to the loss are obtained by solving:

$$\left(\lambda \mathbf{I} + (L-1)\,\hat{A}\right) \sum_i \frac{\partial \mathcal{L}}{\partial \mathbf{b}_i} = \sum_i \frac{\partial \mathcal{L}}{\partial \mathbf{x}_i}, \quad (9) \qquad (\lambda \mathbf{I} - \hat{A}) \frac{\partial \mathcal{L}}{\partial \mathbf{b}_k} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_k} - \hat{A} \sum_i \frac{\partial \mathcal{L}}{\partial \mathbf{b}_i}. \quad (10)$$

Finally, the gradients of $\hat{A}$ are computed as

$$\frac{\partial \mathcal{L}}{\partial \hat{A}} = \frac{\partial \mathcal{L}}{\partial \mathbf{b}_k} \otimes \sum_{i \neq k} \mathbf{x}_i. \quad (11)$$

Thus, rather than solving a system with $A \in \mathbb{R}^{PL \times PL}$, we solve $L+1$ systems with $\hat{A} \in \mathbb{R}^{P \times P}$. In our case, where $L = 21$ for 20 object classes and 1 background class, this simplification empirically reduces the inference time by a factor of 6, and the overall training time by a factor of 3. We expect even larger acceleration for the MS-COCO dataset which has 80 semantic classes. Despite this simplification, the results are competitive to the general setting as shown in Sec. 4.

3 Linear Systems for Efficient and Effective Structured Prediction

Having identified that both the inference problem in Eq. 2 and computation of pairwise gradients in Eq. 3 require the solution of a linear system of equations, we now discuss methods for accelerated inference that rely on standard numerical analysis techniques for linear systems [20,21]. Our main contributions consist in (a) using fast linear system solvers that exhibit fast convergence (Sec. 3.1) and (b) performing inference on multi-scale graphs by constructing block-structured linear systems (Sec. 3.2).

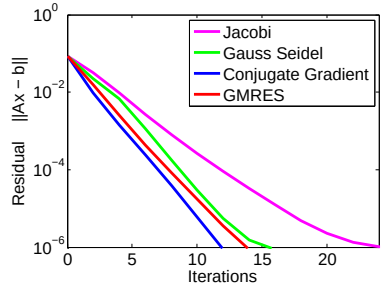
Our contributions in (a) indicate that standard conjugate gradient based linear system solvers can be up to 2.5 faster than the solutions one could get by a naive application of parallel mean-field when implemented on the GPU. Our contribution in (b) aims at accuracy rather than efficiency, and is experimentally validated in Sec. 4

3.1 Fast Linear System Solvers

The computational cost of solving the linear system of equations in Eq. 2 and Eq. 3 depends on the size of the matrix A, i.e. $N \times N$, and its sparsity pattern. In our experiments, while $N \sim 10^5$, the matrix A is quite sparse, since we deal with small

Method	Iterations
Jacobi	24.8
Gauss Siedel	16.4
GMRES	14.8
Conjugate Gradient	13.2

(a) Linear Solver Statistics



(c) Iterative Solvers Convergence

Fig. 2: The table in (a) shows the average number of iterations required by various algorithms, namely Jacobi, Gauss Seidel, Conjugate Gradient, and Generalized Minimal Residual (GMRES) iterative methods to converge to a residual of tolerance 10^{-6}. Figure (b) shows a plot demonstrating the convergence of these iterative solvers. The conjugate gradient method outperforms the other competitors in terms of number of iterations taken to converge.

$4-$connected, $8-$connected and $12-$connected neighbourhoods. While a number of direct linear system solver methods exist, the sheer size of the system matrix A renders them prohibitive, because of large memory requirements. For large problems, a number of iterative methods exist, which require less memory, come with convergence (to a certain tolerance) guarantees under certain conditions, and can be faster than direct methods. In this work, we considered the *Jacobi, Gauss-Seidel, Conjugate Gradient, and Generalized Minimal Residual* (GMRES) methods [20], as candidates for iterative solvers. The table in Fig. 2 (a) shows the average number of iterations required by the aforementioned methods for solving the inference problem in Eq. 2. We used 25 images in this analysis, and a tolerance of 10^{-6}. Fig. 2 shows the convergence of these methods for one of these images. Conjugate gradients clearly stand out as being the fastest of these methods, so our following results use the conjugate gradient method. Our findings are consistent with those of Grady in [22].

As we show below, mean-field inference for the Gaussian CRF can be understood as solving the linear system of Eq. 2, namely parallel mean-field amounts to using the Jacobi algorithm while sequential mean-field amounts to using the Gauss-Seidel algorithm, which are the two weakest baselines in our comparisons. This indicates that by resorting to tools for solving linear systems we have introduced faster alternatives to those suggested by mean field.

In particular the *Jacobi* and *Gauss-Seidel* methods solve a system of linear equations $A\mathbf{x} = B$ by generating a sequence of approximate solutions $\left\{\mathbf{x}^{(k)}\right\}$, where the current solution $\mathbf{x}^{(k)}$ determines the next solution $\mathbf{x}^{(k+1)}$.

The update equation for the *Jacobi* method [23] is given by

$$x_i^{(k+1)} \leftarrow \frac{1}{a_{ii}} \left\{ b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right\}. \tag{12}$$

The updates in Eq. 12 only use the previous solution $\mathbf{x}^{(k)}$, ignoring the most recently available information. For instance, $x_1^{(k)}$ is used in the calculation of $x_2^{(k+1)}$, even though $x_1^{(k+1)}$ is known. This allows for parallel updates for $\mathbf{x}$. In contrast, the *Gauss-Seidel* [23] method always uses the most current estimate of x_i as given by:

$$x_i^{(k+1)} \leftarrow \frac{1}{a_{ii}} \left\{ b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right\}. \tag{13}$$

As in [24], the Gaussian Markov Random Field (GMRF) in its canonical form is expressed as $\pi(\mathbf{x}) \propto \exp\left\{\frac{1}{2}\mathbf{x}^T\Theta\mathbf{x} + \theta^T\mathbf{x}\right\}$, where θ and Θ are called the canonical parameters associated with the multivariate Gaussian distribution $\pi(\mathbf{x})$. The update equation corresponding to mean-field inference is given by [25],

$$\mu_i \leftarrow -\frac{1}{\Theta_{ii}} \left\{ \theta_i + \sum_{j \neq i} \Theta_{ij} \mu_j \right\}, \tag{14}$$

The expression in Eq. 14 is exactly the expression for the *Jacobi* iteration (Eq. 12), or the *Gauss-Seidel* iteration in Eq. 13 for solving the linear system $\mu = -\Theta^{-1}\theta$, depending on whether we use sequential or parallel updates.

One can thus understand sequential and parallel mean-field inference and learning algorithms as relying on weaker system solvers than the conjugate gradient-based ones we propose here. The connection is accurate for Gaussian CRFs, as in our work and [2], and only intuitive for Discrete CRFs used in [1,3].

3.2 Multiresolution graph architecture

We now turn to incorporating computation from multiple scales in a single system. Even though CNNs are designed to be largely scale-invariant, it has been repeatedly reported [26,27] that fusing information from a CNN operating at multiple scales can improve image labeling performance. These results have been obtained for feedforward CNNs - we consider how these could be extended to CNNs with lateral connections, as in our case. A simple way of achieving this would be to use multiple image resolutions, construct one structured prediction module per resolution, train these as disjoint networks, and average the final results. This amounts to solving three decoupled systems which by itself yields a certain improvement as reported in Sec. 4

We advocate however a richer connectivity that couples the scale-specific systems, allowing information to flow across scales. As illustrated in Fig. 3 the resulting linear system captures the following multi-resolution interactions simultaneously: (a) pairwise constraints between pixels at each resolution, and (b) pairwise constraints between the same image region at two different resolutions. These inter-resolution pairwise terms connect a pixel in the image at one resolution, to the pixel it would spatially correspond to at another resolution. The inter-resolution connections help enforce a different kind of pairwise consistency: rather than encouraging pixels in a neighbourhood to have the same/different label, these encourage image regions to have the same/different labels across resolutions. This is experimentally validated in Sec. 4 to outperform the simpler multi-resolution architecture outlined above.

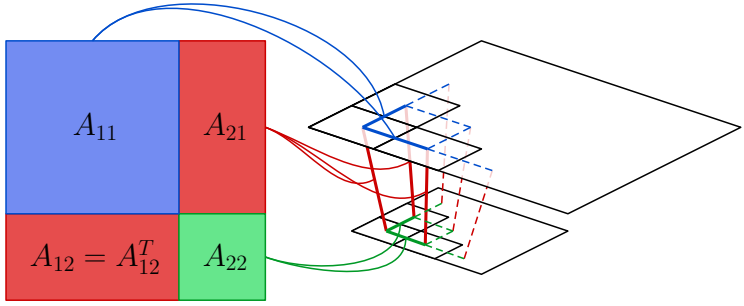


Fig. 3: Schematic diagram of matrix A for the multi-resolution formulation in Sec. 3.2. In this example, we have the input image at 2 resolutions. The pairwise matrix A contains two kinds of pairwise interactions: (a) neighbourhood interactions between pixels at the same resolution (these interactions are shown as the blue and green squares), and (b) interactions between the same image region at two resolutions (these interactions are shown as red rectangles). While interactions of type (a) encourage the pixels in a neighbourhood to take the same or different label, the interactions of type (b) encourage the same image region to take the same labels at different resolutions.

3.3 Implementation Details and Computational Efficiency

Our implementation is fully GPU based, and implemented using the *Caffe* library. Our network processes input images of size 865×673, and delivers results at a resolution that is 8 times smaller, as in [3]. The input to our QO modules is thus a feature map of size 109×85. While the testing time per image for our methods is between $0.4 - 0.7$s per image, our inference procedure only takes $\sim$ 0.02s for the general setting in Sec. 2, and 0.003s for the simplified formulation (Sec. 2.5). This is significantly faster than dense CRF postprocessing, which takes 2.4s for a 375×500 image on a CPU and the 0.24s on a GPU. Our implementation uses the highly optimized *cuBlas* and *cuSparse* libraries for linear algebra on large sparse matrices. The *cuSparse* library requires the matrices to be in the compressed-storage-row (CSR) format in order to fully optimize linear algebra for sparse matrices. Our implementation caches the indices of the CSR matrices, and as such their computation time is not taken into account in the calculations above, since their computation time is zero for streaming applications, or if the images get warped to a canonical size. In applications where images may be coming at different dimensions, considering that the indexes have been precomputed for the changing dimensions, an additional overhead of ~ 0.1s per image is incurred to read the binary files containing the cached indexes from the hard disk (using an SSD drive could further reduce this). Our code and experiments are publicly available at https://github.com/siddharthachandra/gcrf.

4 Experiments

In this section, we describe our experimental setup, network architecture and results.

Dataset. We evaluate our methods on the *VOC PASCAL 2012 image segmentation benchmark*. This benchmark uses the VOC PASCAL 2012 dataset, which consists of 1464 training and 1449 validation images with manually annotated pixel-level labels for 20 foreground object classes, and 1 background class. In addition, we exploit the additional pixel-level annotations provided by [6], obtaining 10582 training images in total. The test set has 1456 unannotated images. The evaluation criterion is the pixel intersection-over-union (IOU) metric, averaged across the 21 classes.

Baseline network (basenet). Our basenet is based on the Deeplab-LargeFOV network from [3]. As in [27], we extend it to get a multi-resolution network, which operates at three resolutions with tied weights. More precisely, our network downsamples the input image by factors of 2 and 3 and later *fuses* the downsampled activations with the original resolution via concatenation followed by convolution. The layers at three resolutions share weights. This acts like a strong baseline for a purely feedforward network. Our basenet has 49 convolutional layers, 20 pooling layers, and was pretrained on the MS-COCO 2014 trainval dataset [28]. The initial learning rate was set to 0.01 and decreased by a factor of 10 at 5K iterations. It was trained for 10K iterations.

QO network. We extend our basenet to accommodate the binary stream of our network. Fig. 1 shows a rough schematic diagram of our network. The basenet forms the unary stream of our QO network, while the pairwise stream is composed by concatenating the 3^{rd} pooling layers of the three resolutions followed by *batch normalization* and two convolutional layers. Thus, in Fig. 1, layers $C_1 - C_3$ are shared by the unary and pairwise streams in our experiments. Like our basenet, the QO networks were trained for 10K iterations; The initial learning rate was set to 0.01 which was decreased by a factor of 10 at 5K iterations. We consider three main types of QO networks: plain (QO), shared weights (QO^s) and multi-resolution (QO^{mres}).

4.1 Experiments on train+aug - val data

In this set of experiments we train our methods on the *train+aug* images, and evaluate them on the *val* images. All our images were upscaled to an input resolution of 865×673. The hyper-parameter λ was set to 10 to ensure positive definiteness. We first study the effect of having larger neighbourhoods among image regions, thus allowing richer connectivity. More precisely, we study three kinds of connectivities: (a) $4-$connected (QO_4), where each pixel is connected to its left, right, top, and bottom neighbours, (b) $8-$connected (QO_8), where each pixel is additionally connected to the 4 diagonally adjacent neighbours, and (c) $12-$connected (QO_{12}), where each pixel is connected to 2 left, right, top, bottom neighbours besides the diagonally adjacent ones. Table 1 demonstrates that while there are improvements in performance upon increasing connectivities, these are not substantial. Given that we obtain diminishing returns, rather than trying even larger neighbourhoods to improve performance, we focus on increasing the richness of the representation by incorporating information from various scales. As described in Sec. 3.2, there are two ways to incorporate information from multiple scales; the simplest is to have one QO unit per resolution (QO^{res}), thereby enforcing pairwise consistencies individually at each resolution before fusing them, while the more sophisticated one is to have information flow both within and across scales, amounting to a joint multi-scale CRF inference task, illustrated in Fig. 3. In Table 2, we

Method	QO_4	QO_8	QO_{12}
IoU	76.36	76.40	76.42

Table 1: Connectivity

Method	QO	QO^s	QO^{res}	QO^{mres}
IoU	76.36	76.59	76.69	76.93

Table 2: Comparison of 4 variants of our G-CRF network.

Method	IoU	IoU after *Dense CRF*
Basenet	72.72	73.78
QO	73.41	75.13
QO^s	73.20	75.41
QO^{mres}	73.86	75.46

Table 3: Performance of our methods on the VOC PASCAL 2012 Image Segmentation Benchmark. Our baseline network (Basenet) is a variant of Deeplab-LargeFOV [3] network. In this table, we demonstrate systematic improvements in performance upon the introduction of our Quadratic Optimization (QO), and multi-resolution (QO^{mres}) approaches. DenseCRF post-processing gives a consistent boost in performance.

compare 4 variants of our QO network: (a) QO (Sec. 2), (b) QO with shared weights (Sec. 2.5), (c) three QO units, one per image resolution, and (d) multi-resolution QO (Sec. 3.2). It can be seen that our weight sharing simplification, while being significantly faster, also gives better results than QO. Finally, the multi-resolution framework outperforms the other variants, indicating that having information flow both within and across scales is desirable, and a unified multi-resolution framework is better than merely averaging QO scores from different image resolutions.

4.2 Experiments on train+aug+val - test data

In this set of experiments, we train our methods on the *train+aug+val* images, and evaluate them on the *test* images. The image resolutions and λ values are the same as those in Sec. 4.1. In these experiments, we also use the Dense CRF post processing as in [3,29]. Our results are tabulated in Tables 3 and 4. We first compare our methods QO, QO^s and QO^{mres} with the basenet, where the relative improvements can be most clearly demonstrated. Our multi-resolution network outperforms the basenet and other QO networks. We achieve a further boost in performance upon using the Dense CRF post processing strategy, consistently for all methods. We observe that our method yields an improvement that is entirely complementary to the improvement obtained by combining with Dense-CRF.

We also compare our results to previously published benchmarks in Table 4. When benchmarking against directly comparable techniques, we observe that even though we do not use end-to-end training for the CRF module stacked on top of our QO network, our method outperforms the previous state of the art CRF-RNN system of [1] by a margin of 0.8%. We anticipate further improvements by integrating end-to-end CRF training with our QO. In Table 4, we compare our methods to previously published, directly comparable methods, namely those that use a variant of the VGG [30] network, are

trained in an end-to-end fashion, and use structured prediction in a fully-convolutional framework. Please note that using deep-residual-networks [31], the recently released Deeplab-V2 [32] has pushed the state of the art to 79.7 mean IoU, outperforming the previous state of the art methods [14,15]. We are working on using our approach in conjunction with Deeplab-V2.

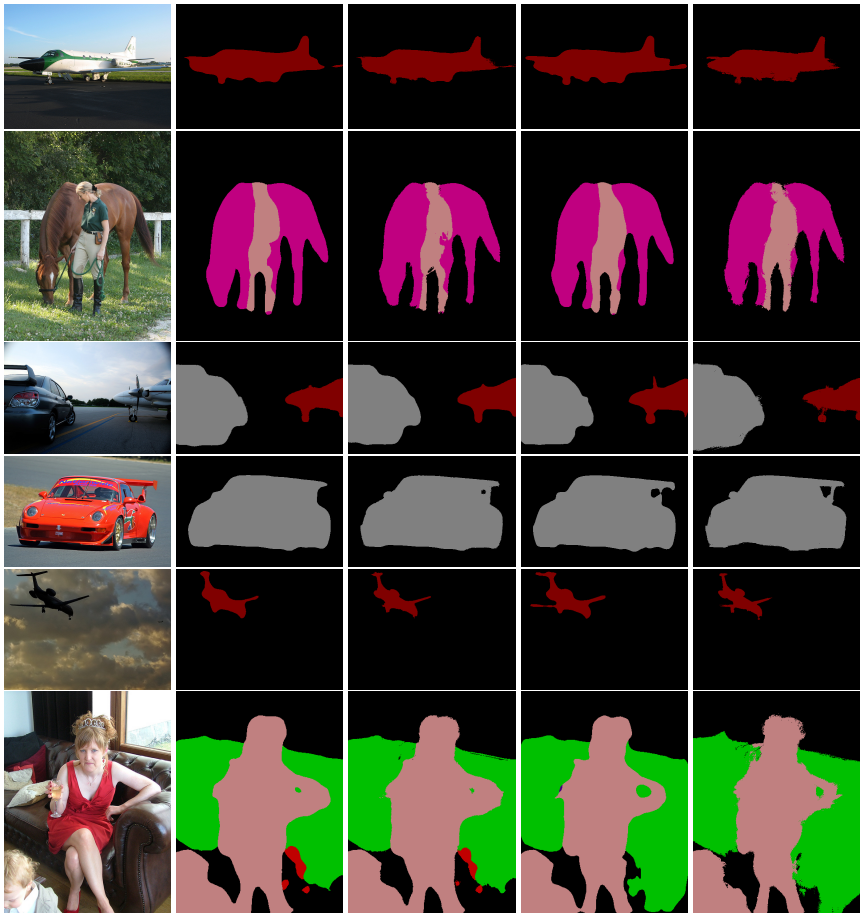
Method	mean IoU (%)
Deeplab-Cross-Joint [29]	73.9
CRFRNN [1]	74.7
Basenet	73.8
QO	75.1
QO^s	75.4
QO^{mres}	75.5

Table 4: Comparison of our method with directly comparable previously published approaches on the VOC PASCAL 2012 image segmentation benchmark.

5 Conclusions and Future Work

In this work we propose a quadratic optimization method for deep networks which can be used for predicting continuous vector-valued variables. The inference is efficient and exact and can be solved in 0.02 seconds on the GPU for each image in the general setting, and 0.003 seconds for the Potts-type pairwise case using the conjugate gradient method. We propose a deep-learning framework which learns features and model parameters simultaneously in an end-to-end FCN training algorithm. Our implementation is fully GPU based, and implemented using the *Caffe* library. Our experimental results indicate that using pairwise terms boosts performance of the network on the task of image segmentation, and our results are competitive with the state of the art methods on the VOC 2012 benchmark, while being substantially simpler. While in this work we focused on simple $4-12$ connected neighbourhoods, we would like to experiment with fully connected graphical models. Secondly, while we empirically verified that setting a constant λ parameter brought about positive-definiteness, we are now exploring approaches to ensure this constraint in a general case. We intend to exploit our approach for solving other regression and classification tasks as in [33,34]. We are currently working on applying our models in conjunction with ResNets [31] as in [32] and will be making our code publicly available.

Acknowledgements This work has been funded by the EU Projects MOBOT FP7-ICT-2011-600796 and I-SUPPORT 643666 #2020.



(a) Image (b) Basenet (c) Basenet + DCRF (d) QO^{mres} (e) QO^{mres} + DCRF

Fig. 4: Visual results on the VOC PASCAL 2012 test set. The first column shows the colour image, the second column shows the basenet predicted segmentation, the third column shows the basenet output after Dense CRF post processing. The fourth column shows the QO^{mres} predicted segmentation, and the final column shows the QO^{mres} output after Dense CRF post processing. It can be seen that our multi-resolution network captures the finer details better than the basenet: the tail of the airplane in the first image, the person's body in the second image, the aircraft fan in the third image, the road between the car's tail in the fourth image, and the wings of the aircraft in the final image, all indicate this. While Dense CRF post-processing quantitatively improves performance, it tends to miss very fine details.

References

1. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.: Conditional random fields as recurrent neural networks. In: ICCV. (2015) 1, 2, 3, 9, 12, 13
2. Vemulapalli, R., Tuzel, O., Liu, M.Y., Chellapa, R.: Gaussian conditional random field network for semantic segmentation. In: CVPR. (June 2016) 1, 2, 3, 9
3. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected crfs. arXiv preprint arXiv:1412.7062 (2014) 1, 2, 3, 9, 10, 11, 12
4. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. PAMI (2013) 1
5. Mostajabi, M., Yadollahpour, P., Shakhnarovich, G.: Feedforward semantic segmentation with zoom-out features. In: CVPR. (2015) 1
6. Hariharan, B., Arbeláez, P., Girshick, R., Malik, J.: Hypercolumns for object segmentation and fine-grained localization. In: CVPR. (2015) 1, 11
7. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR. (2015) 3431–3440 1
8. Farabet, C., Couprie, C., Najman, L., Lecun, Y.: Scene parsing with multiscale feature learning, purity trees, and optimal covers. In: ICML. (2012) 1
9. Chen, L.C., Schwing, A.G., Yuille, A.L., Urtasun, R.: Learning Deep Structured Models. In: ICML. (2015) 1, 2
10. Vemulapalli, R., Tuzel, O., Liu, M.: Deep gaussian conditional random field network: A model-based deep network for discriminative denoising. In: CVPR. (2016) 1, 2, 3
11. Ionescu, C., Vantzos, O., Sminchisescu, C.: Matrix backpropagation for deep networks with structured layers. In: ICCV. (2015) 1
12. Krähenbühl, P., Koltun, V.: Efficient inference in fully connected crfs with gaussian edge potentials. In: NIPS. (2011) 1
13. Couprie, C.: Multi-label energy minimization for object class segmentation. In: Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European, IEEE (2012) 2233–2237 2
14. Lin, G., Shen, C., Reid, I.D., van den Hengel, A.: Efficient piecewise training of deep structured models for semantic segmentation. CVPR (2016) 2, 3, 13
15. Liu, Z., Li, X., Luo, P., Loy, C.C., Tang, X.: Semantic image segmentation via deep parsing network. In: CVPR. (2015) 1377–1385 3, 13
16. Tappen, M.F., Liu, C., Adelson, E.H., Freeman, W.T.: Learning gaussian conditional random fields for low-level vision. In: CVPR. (2007) 3
17. Jancsary, J., Nowozin, S., Sharp, T., Rother, C.: Regression tree fields - an efficient, non-parametric approach to image labeling problems. In: CVPR. (2012) 3, 5
18. Vu, T.H., Osokin, A., Laptev, I.: Context-aware cnns for person head detection. In: ICCV. (2015) 2893–2901 3
19. Shewchuk, J.R.: An introduction to the conjugate gradient method without the agonizing pain. https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf 5
20. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C, 2nd Edition. Cambridge University Press (1992) 7, 8
21. Golub, G.H., Loan, C.F.V.: Matrix computations (3. ed.). Johns Hopkins University Press (1996) 7
22. Grady, L.: Random walks for image segmentation. In: PAMI. (2006) 8
23. Golub, G.H., Loan, V., F., C.: Matrix computations. **3**(1-2) (January 1996) 510 8, 9

24. Rue, H., Held, L.: Gaussian Markov Random Fields: Theory and Applications. Volume 104 of Monographs on Statistics and Applied Probability. Chapman & Hall, London (2005) 9
25. Wainwright, M.J., Jordan, M.I.: Graphical models, exponential families, and variational inference. Found. Trends Mach. Learn. **1**(1-2) (January 2008) 136–138 9
26. Chen, L., Yang, Y., Wang, J., Xu, W., Yuille, A.L.: Attention to scale: Scale-aware semantic image segmentation. CVPR (2016) 9
27. Kokkinos, I.: Pushing the Boundaries of Boundary Detection using Deep Learning. In: ICLR. (2016) 9, 11
28. *et al*., L.: Microsoft coco: Common objects in context. In: ECCV. (2014) 11
29. Chen, L.C., Papandreou, G., Murphy, K., Yuille, A.L.: Weakly- and semi-supervised learning of a deep convolutional network for semantic image segmentation. ICCV (2015) 12, 13
30. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. ICLR (2015) 12
31. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. (2016) 13
32. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. arXiv:1606.00915 (2016) 13
33. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: ICCV. (2015) 2650–2658 13
34. Kokkinos, I.: Ubernet: A universal cnn for the joint treatment of low-, mid-, and high- level vision problems. In: POCV workshop. (2016) 13