# Deep Spatio-Temporal Random Fields for Efficient Video Segmentation

Siddhartha Chandra[1] siddhartha.chandra@inria.fr
Camille Couprie[2] coupriec@fb.com
Iasonas Kokkinos[2] iasonask@fb.com
[1] INRIA GALEN, Ecole CentraleSupélec Paris
[2] Facebook AI Research, Paris

## Abstract

*In this work we introduce a time- and memory-efficient method for structured prediction that couples neuron decisions across both space at time. We show that we are able to perform exact and efficient inference on a densely-connected spatio-temporal graph by capitalizing on recent advances on deep Gaussian random fields. We experiment with multiple connectivity patterns in the temporal domain, and present empirical improvements over strong baselines on the tasks of both semantic and instance segmentation of videos. Our proposed approach is (a) efficient, (b) has a unique global minimum, and (c) can be trained end-to-end alongside contemporary deep networks for video understanding. Our implementation is based on the* Caffe2 *framework and will be made publicly available.*

## 1. Introduction

Video understanding remains largely unsolved despite dramatic improvements in image understanding over the past five years. The accuracy of current image classification, or semantic segmentation models is not yet matched in action recognition and video segmentation, to some extent due to the lack of large-scale benchmarks, but also due to the complexity introduced by introducing the time variable. Combined with increase in memory and computation demands, video understanding poses additional challenges that call for novel methods.

Our objective in this work is to go beyond the frame-by-frame processing currently used in most CNN-based architectures. We focus on coupling the decisions taken by a neural network in time, in a manner that allows information to flow across frames resulting in decisions that are consistent both spatially and temporally. Towards this goal we pursue a structured prediction approach, where the structure of the output space is exploited in order to train classifiers of higher accuracy. For this we introduce into video segmentation the Deep Gaussian Random Field (DGRF) technique recently proposed for single-frame structured prediction in [5, 6]. Our main technical contribution consists in adapting this method so that it becomes affordable for video segmentation, both from a time- and memory- complexity viewpoint. To this end, we propose a customized conjugate gradient method that eliminates redundant computations by exploiting the structure of the temporal neighbourhood.

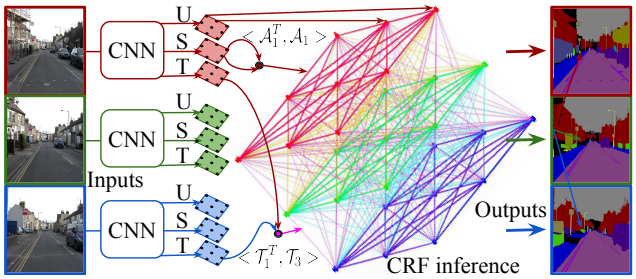



Figure 1. Overview of our approach: our deep network takes in $V$ input video frames and delivers unary terms (U), spatial (S) and temporal (T) embeddings for each of the frames to a dense G-CRF module. The G-CRF module uses the unary terms and spatio-temporal embeddings to express the unary, the intra- and inter-frame pairwise terms for a densely connected Gaussian-CRF. The dense G-CRF module does spatio-temporal structured prediction via efficient G-CRF inference to deliver output which is used to generate segmentations. Our network can be trained in an end-to-end manner. Please note that the 3 frames in this example are colour coded as red, green, and blue: these colours are used to indicate correspondences between the inputs, network outputs, and unary, pairwise interactions in the G-CRF.

We show that our algorithm can be used for a variety of video segmentation tasks: semantic segmentation (CamVid dataset), instance tracking (DAVIS dataset), and a combination of instance segmentation with Mask-RCNN-style object detection, customized in particular for the person class (DAVIS Person dataset).

Our work inherits all favorable properties of the DGRF method: in particular, our method has the advantage of delivering (a) exact inference results through the solution of a linear system, rather than relying on approximate mean-field inference, as [25, 26], (b) allowing for exact computation of the gradient during back-propagation,

thereby alleviating the need for the memory-demanding back-propagation-through-time used in [42] (c) making it possible to use non-parametric terms for the pairwise term, rather than confining ourselves to pairwise terms of a predetermined form, as [25, 26], and facilitating inference on both densely- and sparsely-connected graphs, as well as facilitating blends of both graph topologies.

Within the literature on spatio-temporal structured prediction, the work that is closest in spirit to ours is the work of [26] on Feature Space Optimization. Even though our works share several conceptual similarities, our method is entirely different at the technical level in the sense that it is conceived as a neural network module for structured prediction that is trained jointly with the convnets that process the individual frames, while the method of [26] is applied at a post-processing stage to refine a classifier's results.

### 1.1. Previous work

Structured prediction is commonly used by semantic segmentation algorithms [5, 6, 7, 9, 10, 36, 39, 42] to capture spatial constraints within an image frame. These approaches may be extended naively to videos, by making predictions individually for each frame. However, in doing so, we ignore the temporal context, thereby ignoring the tendency of consecutive video frames to be similar to each other. To address this shortcoming, a number of deep learning methods employ some kind of structured prediction strategy to ensure temporal coherence in the predictions. Initial attempts to capturing spatio-temporal context involved designing deep learning architectures [23] that implicitly learnt interactions between consecutive image frames. A number of subsequent approaches used Recurrent Neural Networks (RNNs) [2, 12] to capture interdependencies between the image frames. Further, several approaches have exploited optical flow computed from state of the art approaches, as in [18], as additional input to the network [15, 19]. Finally, methods that rely on explicit capturing of these temporal constraints via pairwise terms over probabilistic graphical models also exist [3, 26].

In this work, we focus on three problems, namely (i) semantic and (ii) instance video segmentation, and (iii) semantic instance tracking. Semantic instance tracking refers to the problem where we are given the ground truth for the first frame of a video, and the goal is to predict these instance masks on the subsequent frames of the video. Contemporary deep learning literature describes two distinct approaches to this task. The first set of approaches start with a deep network pretrained for image classification on large datasets such as Imagenet or COCO, and finetune it on the first frame of the video with labeled ground truth [4, 38], optionally leveraging a variety of data augmentation regimes [24] to increase robustness to scale/pose variation and occlusion/truncation in the subsequent frames of the video. The second set of approaches poses this problem as a warping problem [30], where the goal is to warp the segmentation of the first frame using the images and optical flow as additional inputs [20, 24, 27].

A number of approaches have attempted to exploit temporal information to improve over static image segmentation approaches for video segmentation. Clockwork convnets [33] were introduced to exploit the persistence of features across time and schedule the processing of some layers at different update rates according to their semantic stability. Similar feature flow propagation ideas were employed in [26, 43]. In [29], the segmentations are warped using the flow and spatial transformer networks. Rather than using optical flow, the prediction of future segmentations [22] may also temporally smooth frame by frame results. Finally, the state-of-the-art improving over PSPnet[41] is achieved by warping the feature maps of a static segmentation CNN to emulate a video segmentation network [15].

## 2. Spatio-Temporal Gaussian Random Fields

In this work we extend the Deep Gaussian CRF approach introduced in [5, 6] to operate efficiently for video segmentation. Introducing a CRF allows us to couple the decisions between sets of variables that should be influencing each other; spatial connections were already explored in [5, 6] and can be understood as propagating information from distinctive image positions (e.g. the face of a person) to more ambiguous regions (e.g. the person's clothes). In this work we also introduce temporal connections, which allow us to smooth information over time, and for instance to correctly segment frames where the object is not clearly visible by propagating information from different time frames.

We consider that the input to our system is a video $\mathcal{V} = \{I_1, I_2, \ldots, I_V\}$ containing $V$ frames. We denote our network's prediction as $\mathbf{x}_v$, $v = 1, \ldots, V$, where at any frame the prediction $\mathbf{x}_i \in \mathbb{R}^{PL}$ provides a real valued vector giving a distribution of scores over the $L$ classes for each of the $P$ image patches; for brevity, we denote by $N = P \times L$ the number of prediction variables. The $L$ scores corresponding to a patch can be understood as inputs to a softmax function that yields the label posteriors.

Given a video input, our G-CRF model defines a joint posterior distribution through a Gaussian multivariate density:

$$p(\mathbf{x}|\mathcal{V}) \propto \exp(-\frac{1}{2}\mathbf{x}^\top A_\mathcal{V}\mathbf{x} + B_\mathcal{V}\mathbf{x}),$$

where $B_\mathcal{V}$, $A_\mathcal{V}$ denote the 'unary' and 'pairwise' terms respectively, with $B_\mathcal{V} \in \mathbb{R}^{NV}$ and $A_\mathcal{V} \in \mathbb{R}^{NV \times NV}$. In the rest of this work, it is assumed that the parameters $A, B$ depend on the input video, and we omit the conditioning on $\mathcal{V}$ for notational convenience.

What is particular about the G-CRF is that, assuming the matrix of pairwise terms $A$ is positive-definite, the

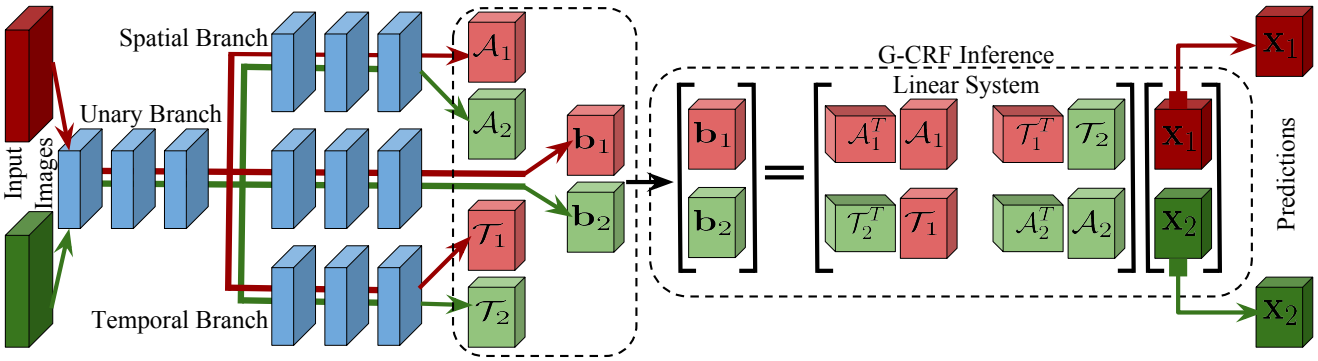



Figure 2. Spatio-Temporal G-CRF schematic for 2 video-frames. Our network takes in two input images, and delivers the per frame unaries $\mathbf{b}_1, \mathbf{b}_2$, spatial embeddings $\mathcal{A}_1, \mathcal{A}_2$, and temporal embeddings $\mathcal{T}_1, \mathcal{T}_2$ in the feed-forward mode. Our spatio-temporal G-CRF module collects these and solves the inference problem described in Eq. 2 to recover predictions $\mathbf{x}_1, \mathbf{x}_2$ for the two frames. During backward pass, the gradients of the predictions are delivered to the spatio-temporal G-CRF model. It uses these to compute the gradients for the unary terms as well as the spatio-temporal embeddings and back-propagates them through the network.

Maximum-A-Posterior (MAP) inference merely amounts to solving the system of linear equations $A\mathbf{x} = B$. In fact, as in [5], we can drop the probabilistic formulation and treat the G-CRF as a structured prediction module that is part of a deep network. In the forward pass, the unary and the pairwise terms $B$ and $A$, delivered by a feed-forward CNN described in Sec. 2.1, are fed to the G-CRF module which performs inference to recover the prediction $\mathbf{x}$ by solving a system of linear equations given by

$$(A + \lambda \mathbf{I})\mathbf{x} = B, \tag{1}$$

where $\lambda$ is a small positive constant added to the diagonal entries of $A$ to make it positive definite.

For the single-frame case ($V = 1$) the iterative conjugate gradient [34] algorithm was used to rapidly solve the resulting system for both sparse [5] and fully connected [6] graphs; in particular the speed of the resulting inference is in the order of 30ms on the GPU, almost two orders of magnitude faster than the implementation of DenseCRF, while at the same time giving more accurate results.

Our first contribution in this work consists in designing the structure of the matrix $A_{\mathcal{V}}$ so that the resulting system solution remains manageable as the number of frames increases. Once we describe how we structure $A_{\mathcal{V}}$, we then will turn to learning our network in an end-to-end manner.

## 2.1. Spatio-temporal connections

In order to capture the spatio-temporal context, we are interested in capturing two kinds of pairwise interactions: (a) pairwise terms between patches in the same frame,and (b) pairwise terms between patches in different frames.

Denoting the spatial pairwise terms at frame $v$ by $A_v$ and the temporal pairwise terms between frames $u, v$ as $T_{u,v}$ we can rewrite Eq. 1 as follows:

$$\begin{bmatrix} A_1 + \lambda \mathbf{I} & T_{1,2} & \cdots & T_{1,V} \\ T_{2,1} & A_2 + \lambda \mathbf{I} & \cdots & T_{2,V} \\ & & \vdots & \\ T_{V,1} & T_{V,2} & \cdots & A_V + \lambda \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_V \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_V \end{bmatrix}, \tag{2}$$

where we group the variables by frames. Solving this system allows us to couple predictions $\mathbf{x}_v$ across all video frames $v \in \{1, \ldots, V\}$, positions, $p$ and labels $l$. If furthermore $A_v = A_v^T, \forall v$ and $T_{u,v} = T_{v,u}^T, \forall u, v$ then the resulting system is positive definite for any positive $\lambda$.

We start by describing how the pairwise terms $A_v, T_{u,v}$ are constructed through our CNN, and then turn to how the solution of the linear system in Eq. 2 can be accelerated by exploiting its structure.

**Spatial Connections:** We define the spatial pairwise terms in terms of inner products of pixel-wise embeddings, following [6]. At frame $v$ we couple the scores for a pair of patches $p_i, p_j$ taking the labels $l_m, l_n$ respectively as follows:

$$A_{v,p_i,p_j}(l_m, l_n) = \langle \mathcal{A}_{v,p_i}^{l_m}, \mathcal{A}_{v,p_j}^{l_n} \rangle, \tag{3}$$

where $i, j \in \{1, \ldots, P\}$ and $m, n \in \{1, \ldots, L\}$, $v \in \{1, \ldots, V\}$, and $\mathcal{A}_{v,p_j}^{l_n} \in \mathbb{R}^D$ is the embedding associated to point $p_j$. In Eq. 3 the $\mathcal{A}_{v,p_j}^{l_n}$ terms are image-dependent and delivered by a fully-convolutional "embedding" branch that feeds from the same CNN backbone architecture, and is denoted by $\mathcal{A}_v$ in Fig. 2.

The implication of this form is that we can afford inference with a fully-connected graph. In particular the rank of the block matrix $A_v = \mathcal{A}_v^\top \mathcal{A}_v$, equals the embedding dimension $D$, which means that both the memory- and time- complexity of solving the linear system drops from

$O(N^2)$ to $O(ND)$, which can be several orders of magnitude smaller. Thus, $\mathcal{A}_v \in \mathbb{R}^{N \times D}$

**Temporal Connections:** Turning to the *temporal* pairwise terms, we couple patches $p_i, p_j$ coming from different frames $u, v$ taking the labels $l_m, l_n$ respectively as

$$T_{u,v,p_i,p_j}(l_m, l_n) = \langle \mathcal{T}_{u,p_i}^{l_m}, \mathcal{T}_{v,p_j}^{l_n} \rangle, \tag{4}$$

where $u, v \in \{1, \ldots, V\}$. The respective embedding terms are delivered by a branch of the network that is separate, temporal embedding network, denoted by $\mathcal{T}_v$ in Fig. 2.

In short, both the spatial pairwise and the temporal pairwise terms are composed as Gram matrices of spatial and temporal embeddings as $A_v = \mathcal{A}_v^\top \mathcal{A}_v$, and $T_{u,v} = \mathcal{T}_u^\top \mathcal{T}_v$. We visualize our spatio-temporal pairwise terms in Fig. 3.

**Spatio-Temporal G-CRFs in Deep Learning:** Our st-G-CRFs can be viewed as generic deep learning modules for spatio-temporal structured prediction, and as such can be plugged in at any stage of a deep learning pipeline: either as the last layer, i.e. classifier, as in our semantic segmentation experiments (Sec. 3.3), or even in the low-level feature learning stage, as in our instance segmentation experiments (Sec. 3.1).

## 2.2. Efficient Conjugate-Gradient Implementation

We now describe an efficient implementation of the conjugate gradient method [34], described in Algorithm 1 that is customized for our spatio-temporal G-CRFs.

**Algorithm 1** Conjugate Gradient Algorithm

1: **procedure** CONJUGATEGRADIENT
2: **Input:** $\mathbf{A}, \mathbf{B}, \mathbf{x}_0$ **Output:** $\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{B}$
3: $\mathbf{r}_0 := \mathbf{B} - \mathbf{A}\mathbf{x}_0$; $\mathbf{p}_0 := \mathbf{r}_0$; $k := 0$
4: repeat
5: $\alpha_k := \frac{\mathbf{r}_k^\top \mathbf{r}_k}{\mathbf{p}_k^\top \mathbf{A} \mathbf{p}_k}$
6: $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$
7: $\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{p}_k$
8: if $\|\mathbf{r}_{k+1}\|$ is sufficiently small, then exit loop
9: $\beta_k := \frac{\mathbf{r}_{k+1}^\top \mathbf{r}_{k+1}}{\mathbf{r}_k^\top \mathbf{r}_k}$
10: $\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$
11: $k := k + 1$
12: end repeat
13: $\mathbf{x} = \mathbf{x}_{k+1}$

The computational complexity of the conjugate gradient algorithm is determined by the computation of the matrix-vector product $\mathbf{q} = A\mathbf{p}$, corresponding to `line :7` of Algorithm 1 (we drop the subscript $k$ for convenience).

We now discuss how to efficiently compute $\mathbf{q}$ in a manner that is customized for this work. In our case, the matrix-vector product $\mathbf{q} = A\mathbf{p}$ is expressed in terms of the spatial ($\mathcal{A}$) and temporal ($\mathcal{T}$) embeddings as follows:

$$\begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_V \end{bmatrix} = \begin{bmatrix} \mathcal{A}_1^T\mathcal{A}_1 + \lambda\mathbf{I} & \mathcal{T}_1^T\mathcal{T}_2 & \cdots & \mathcal{T}_1^T\mathcal{T}_V \\ \mathcal{T}_2^T\mathcal{T}_1 & \mathcal{A}_2^T\mathcal{A}_2 + \lambda\mathbf{I} & \cdots & \mathcal{T}_2^T\mathcal{T}_V \\ & & \vdots & \\ \mathcal{T}_V^T\mathcal{T}_1 & \mathcal{T}_V^T\mathcal{T}_2 & \cdots & \mathcal{A}_V^T\mathcal{A}_V + \lambda\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_V \end{bmatrix} \tag{5}$$

From Eq. 5, we can express $\mathbf{q}_i$ as follows:

$$\mathbf{q}_i = \mathcal{A}_i^T\mathcal{A}_i\mathbf{p}_i + \lambda\mathbf{p}_i + \sum_{j \neq i} \mathcal{T}_i^T\mathcal{T}_j\mathbf{p}_j. \tag{6}$$

One optimization that we exploit in computing $\mathbf{q}_i$ efficiently is that we do not 'explicitly' compute the matrix-matrix products $\mathcal{A}_i^T\mathcal{A}_i$ or $\mathcal{T}_i^T\mathcal{T}_j$. We note that $\mathcal{A}_i^T\mathcal{A}_i\mathbf{p}_i$ can be decomposed into two matrix-vector products as $\mathcal{A}_i^T(\mathcal{A}_i\mathbf{p}_i)$, where the expression in the brackets is evaluated first and yields a vector, which can then be multiplied with the matrix outside the brackets. This simplification alleviates the need to keep $N \times N$ terms in memory, and is computationally cheaper.

Further, from Eq. 6, we note that computation of $\mathbf{q}_i$ requires the matrix-vector product $\mathcal{T}_j\mathbf{p}_j \;\; \forall j \neq i$. A *black-box* implementation would therefore involve redundant computations. We eliminate this redundancy by rewriting Eq. 6 as

$$\mathbf{q}_i = \mathcal{A}_i^T\mathcal{A}_i\mathbf{p}_i + \lambda\mathbf{p}_i + \mathcal{T}_i^T\left((\sum_j \mathcal{T}_j\mathbf{p}_j) - \mathcal{T}_i\mathbf{p}_i\right). \tag{7}$$

This rephrasing allows us to precompute and cache $\sum_j \mathcal{T}_j\mathbf{p}_j$, thereby eliminating redundant calculations.

While so far we have assumed dense connections between the image frames, if we have sparse temporal connections (Sec. 3.1), i.e. each frame is connected to a subset of neighbouring frames in the temporal domain, the linear system matrix $A$ is sparse, and $\mathbf{q}_i$ is written as

$$\mathbf{q}_i = \mathcal{A}_i^T\mathcal{A}_i\mathbf{p}_i + \lambda\mathbf{p}_i + \sum_{j \in \mathcal{N}(i)} \mathcal{T}_i^T\mathcal{T}_j\mathbf{p}_j, \tag{8}$$

where $\mathcal{N}(i)$ denotes the temporal neighbourhood of frame $i$. For very sparse connections caching may not be necessary because these involve little or no redundant computations.

## 2.3. Backward Pass

By virtue of relying on the Gaussian CRF we can get the back-propagation equation for the gradient of the loss with respect to the unary terms, $\mathbf{b}_v$, and the spatial/temporal embedding terms $\mathcal{A}_v, \mathcal{T}_v$ in closed form, thereby sparing us from having to perform back-propagation in time which was needed e.g. in [42] for DenseCRF inference. Following [6], the gradients of the unary terms $\frac{\partial \mathcal{L}}{\partial \mathbf{b}_v}$ are obtained from the

solution of the following system:

$$\begin{bmatrix} A_1+\lambda\mathbf{I} & T_{1,2} & \cdots & T_{1,V} \\ T_{2,1} & A_2+\lambda\mathbf{I} & \cdots & T_{2,V} \\ & & \vdots & \\ T_{V,1} & T_{V,2} & \cdots & A_V+\lambda\mathbf{I} \end{bmatrix} \begin{bmatrix} \frac{\partial\mathcal{L}}{\partial\mathbf{b}_1} \\ \frac{\partial\mathcal{L}}{\partial\mathbf{b}_2} \\ \vdots \\ \frac{\partial\mathcal{L}}{\partial\mathbf{b}_V} \end{bmatrix} = \begin{bmatrix} \frac{\partial\mathcal{L}}{\partial\mathbf{x}_1} \\ \frac{\partial\mathcal{L}}{\partial\mathbf{x}_2} \\ \vdots \\ \frac{\partial\mathcal{L}}{\partial\mathbf{x}_V} \end{bmatrix}. \tag{9}$$

Once these are computed, the gradients of the spatial embeddings can be computed as follows:

$$\frac{\partial\mathcal{L}}{\partial\mathcal{A}_v} = -\left(\frac{\partial\mathcal{L}}{\partial\mathbf{b}_v}\otimes\mathbf{x}_v\right)\left(\left(\mathbf{I}\otimes\mathcal{A}_v^\top\right)+\left(\mathcal{A}_v^\top\otimes\mathbf{I}\right)Q_{D,N}\right) \tag{10}$$

while the gradients of the temporal embeddings are given by the following form:

$$\frac{\partial\mathcal{L}}{\partial\mathcal{T}_v} = -\sum_u\left(\frac{\partial\mathcal{L}}{\partial\mathbf{b}_u}\otimes\mathbf{x}_v\right)\left(\left(\mathbf{I}\otimes\mathcal{T}_u^\top\right)+\left(\mathcal{T}_u^\top\otimes\mathbf{I}\right)Q_{D,N}\right) \tag{11}$$

where $Q_{D,N}$ is a permutation matrix, defined along the lines of [6]; we provide the derivation in the supplement.

### 2.4. Implementation and Inference Time

Our implementation is GPU based and exploits fast *CUDA-BLAS* routines for linear algebra. It is implemented as a module in the Caffe2 library. For spatial and temporal embeddings of size 128, 12 classes (Sec. 3.3), a $321\times321$ input image, and network stride of 8, our $2,3,4$ frame inferences take 0.032s, 0.045s and 0.061s on average respectively. Without the caching procedure described in Sec. 2.2, the 4 frame inference takes 0.080s on average. This is orders of magnitude faster than the dense-CRF method [25] which takes 0.2s on average for spatial CRF. These timing statistics were estimated on a `GTX-1080` GPU.

## 3. Experiments

**Experimental Setup.** We describe the basic setup followed for our experiments. As in [6], we use a $3-$phase training strategy for our methods. We first train the unary network without the spatio-temporal embeddings. We next train the subnetwork delivering the spatio-temporal embeddings with the softmax cross-entropy loss to enforce the following objectives: $A_{p_1,p_2}(l_1,l_2) < A_{p_1,p_2}(l_1'\neq l_1, l_2'\neq l_2)$, and $T_{u,v,p_1,p_2}(l_1,l_2) < T_{u,v,p_1,p_2}(l_1'\neq l_1, l_2'\neq l_2)$, where $l_1,l_2$ are the ground truth labels for pixels $p_1,p_2$. Finally, we combine the unary and pairwise networks, and train them together in end-to-end fashion. For the embedding branches, we use sub-networks of 10 layers each on top of the standard ResNet-101 `conv-4` layer. Unless otherwise stated, we use stochastic gradient descent to train our networks with a momentum of 0.9 and a weight decay of $5e^{-4}$. For segmentation experiments, we use a base-learning rate of $2.5e^{-3}$ for training the unaries, $2.5e^{-4}$ for training the embeddings, and $1e^{-4}$ for finetuning the unary and embeddings together, using a polynomial-decay with power of 0.9. For the instance segmentation network, we use a single stage training for the unary and pairwise streams: we train the network for 16K iterations, with a base learning rate of 0.01 which is reduced to 0.001 after 12K iterations. The weight decay is $1e^{-4}$. For our instance tracking experiments, we use unaries from [38] and do not refine them, rather use them as an input to our network. We employ horizontal flipping and scaling by factors between 0.5 and 1.5 during training/testing for all methods, except in the case of instance segmentation experiments (Sec. 3.1).

**Datasets.** We now describe the datasets used for our experiments.

*DAVIS.* The DAVIS dataset [31] consists of 30 training and 20 validation videos containing 2079 and 1376 frames respectively. Each video comes with manually annotated segmentation masks for foreground object instances.

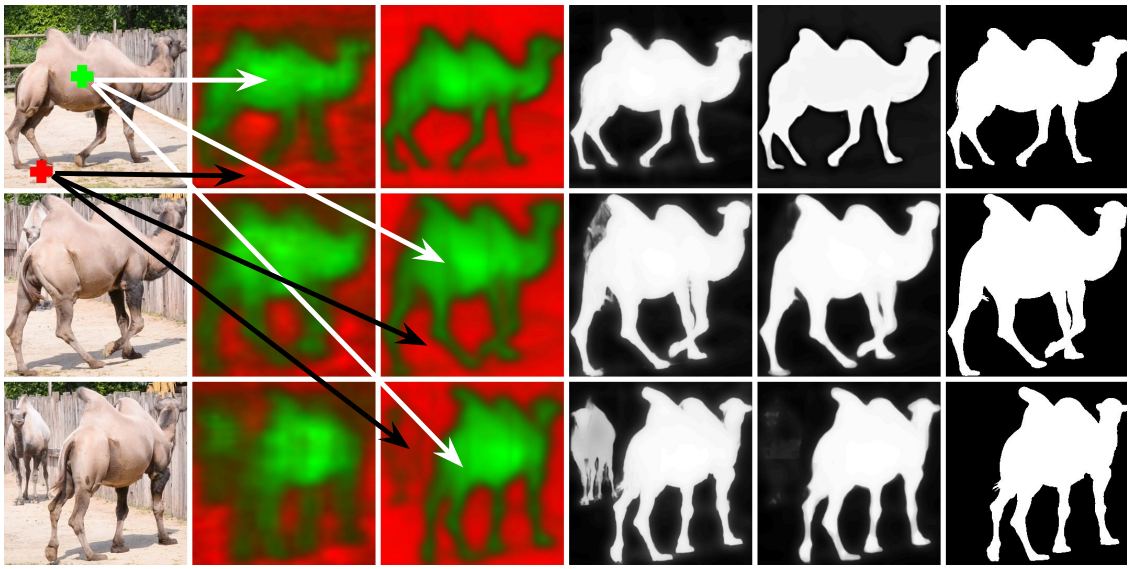
*DAVIS-Person.* While the DAVIS dataset [32] provides densely annotated frames for instance segmentation, it lacks object category labels. For category prediction tasks such as semantic and instance segmentation, we create a subset of the DAVIS dataset containing videos from the category person. By means of visual inspection, we select 35 and 18 video sequences from the training and validation sets respectively containing 2463 training and 1182 validation images, each containing at least one person. Since the DAVIS dataset comes with only the *foreground* instances labeled, we manually annotate the image regions containing *unannotated person* instances with the *do-not-care* label. These image regions do not participate in the training or the evaluation. We call this the DAVIS-person dataset.

*CamVid.* The CamVid dataset [14], is a dataset containing videos of driving scenarios for urban scene understanding. It comes with 701 images annotated with pixel-level category labels at 1 fps. Although the original dataset comes with 32 class-labels, as in [35, 26, 21], we predict 11 semantic classes and use the train-val-test split of 367, 101 and 233 frames respectively.

### 3.1. Ablation Study on Semantic and Instance Segmentation Tasks

In these experiments, we use the DAVIS Person dataset described in Sec. 3. The aim here is to explore the various design choices available to us when designing networks for spatio-temporal structured prediction for semantic segmentation, and proposal based instance segmentation tasks.

**Semantic Segmentation Experiments.** Our first set of experiments studies the effect of varying the sizes of the spatial and temporal embeddings, the degree of the temporal connections, and multi-scale temporal connections for the spatio-temporal G-CRFs. For these set of experiments, our baseline network, or *base-net* is a single resolution ResNet-



**(a) Image** **(b) spatial** **(c) temporal** **(d) unary** **(e) post G-CRF** **(f) GT**

Figure 3. Spatio-temporal G-CRF visualization: We mark two points on the reference frame of the video with + signs. We plot the heatmaps of the spatial and temporal pairwise affinities between these 2 points and all other points on the reference frame (row 1) as well two subsequent frames (rows 2, 3). Further, we show heatmaps of the unary scores, followed by our predictions and the ground truth. We note that while the spatial embeddings may be confused by the presence of a second camel, as in the case of the unary classifier, the temporal context helps recover the correct instance mask.

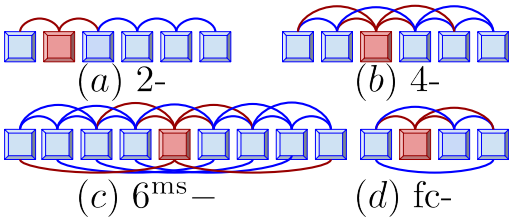



Figure 4. Illustration of the various temporal neighbourhoods we consider in our ablation study. Each box denotes a video frame and the arcs connecting them are pairwise connections. We show a frame in red with all neighbours present in the temporal context.

101 network, with altered network strides as in [8] to produce a spatial down-sampling factor of 8. The network was pretrained on the PASCAL VOC 2012 dataset for semantic segmentation. The evaluation metric used in these experiments is the mean pixel Intersection over Union (IoU).

In Table 1 we study the effect of varying the sizes of the spatial and temporal embeddings, and compare the performance of our methods that couple 2−frames at a time, against that of the base-net. Our best results are achieved at when the sizes of our spatio-temporal embeddings are 128. The improvement over the base-net is 4.2%. In the rest of our experiments in this work, we fix the size of our embeddings to 128. We next study the effect of varying the size of the temporal context and temporal neighbourhoods. The temporal context is defined as the number of video frames $\mathcal{V}$ which are considered simultaneously in one linear system (Eq. 2). The temporal context $\mathcal{V}$ is limited by the size of the network and GPU RAM: for a ResNet-101 network, an input image of size $321 \times 321$, embeddings of size 128, we can currently fit $\mathcal{V} = 7$ frames on 12 GB of GPU RAM, thus the maximum temporal context possible in this setting is 7 frames. Since $\mathcal{V}$ is smaller than the number of frames in the video, we divide the video into overlapping sets of $\mathcal{V}$ frames, and average the predictions for the common frames.

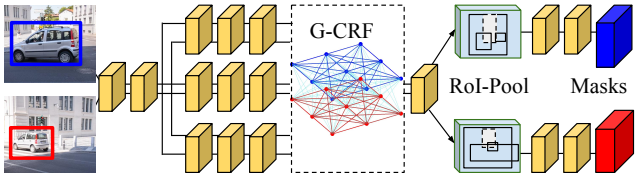



Figure 5. Spatio-temporal structured prediction in Mask-RCNN. Here we use G-CRFs in the feature learning stage before the ROI-Pooling (and not as the final classifier). This helps learn mid-level features which are better aware of the spatio-temporal context.

The temporal neighbourhood for a frame (Fig. 4) is defined as the number of frames it is directly connected to via pairwise connections. A fully connected neighbourhood (fc−) is one in which there are pairwise terms between every pair of frames available in the temporal context. We experiment with 2−, 4−, multiscale $6^{\text{ms}}-$ and fc− connections. The $6^{\text{ms}}-$ neighbourhood connects a frame to neighbours at distances of $2^0$, $2^1$ and $2^2$ (or $1, 2, 4$) frames on either side. Table 2 reports our results for different com-

binations of temporal neighbourhood and context. It can be seen that dense connections improve performance for smaller temporal contexts, but for a temporal context of 7 frames, an increase in the complexity of temporal connections leads to decrease in performance.

| base-net | **81.16** | | | |
|---|---|---|---|---|
| st-G-CRF | spatial dimension→ | | | |
| temporal dimension↓ | 64 | 128 | 256 | 512 |
| 64 | 84.89 | 85.21 | 85.20 | 84.98 |
| 128 | 85.18 | **86.38** | 86.34 | 84.91 |
| 256 | 85.92 | 86.37 | 85.95 | 84.92 |
| 512 | 84.85 | 85.95 | 84.95 | 84.21 |

Table 1. Ablation study: mean IoU on the DAVIS-person dataset using 2 frame fc− connections. Here we study the effect of varying the size of the spatial & temporal embeddings. We fix the size of these embeddings to 128 for subsequent experiments.

| base-net | **81.16** | | | |
|---|---|---|---|---|
| st-G-CRF | temporal neighbourhood → | | | |
| temporal context↓ | 2− | 4− | $6^{ms}$− | fc− |
| 2 | – | – | – | 86.38 |
| 3 | 86.42 | – | – | 86.51 |
| 4 | 86.70 | – | – | 86.82 |
| 7 | **86.98** | 86.79 | 86.82 | 86.42 |

Table 2. Ablation study: mean IoU on the DAVIS-person dataset. Here we study the effect of varying the size of the temporal context and neighbourhood.

**Instance Segmentation Experiments.** In this set of experiments, we demonstrate the utility of our spatio-temporal G-CRF method for the task of proposal based instance segmentation. Our hypothesis here is that coupling of predictions across frames is advantageous for instance segmentation methods, and our goal here is to show that the performance of the instance segmentation methods improves as we increase the temporal context via spatio-temporal G-CRFs: we assume fully connected temporal neighbourhoods. Our baseline for this task is the Mask-RCNN framework of [17] using the ResNet-50 network as the convolutional *body*. The Mask-RCNN framework uses precomputed bounding box proposals for this task. It computes convolutional features on the input image using the convolutional *body* network, crops out the features corresponding to image regions in the proposed bounding boxes via Region-Of-Interest (RoI) pooling, and then has 3 *head* networks to predict (i) class scores and bounding box regression parameters, (ii) keypoint locations, and (iii) instance masks. Structured prediction coupling the predictions of all the proposals over all the video frames is a computationally challenging task, since typically we have $100 - 1000$s of proposals per image, and it is not obvious which proposals from one frame should influence which proposals in the other frame. To circumvent this issue, we use our G-CRFs before the RoI pooling stage as shown in Fig. 5. Thus, rather than coupling final predictions, we are coupling mid-level features over the video frames in an attempt to improve the features which will ultimately be used to make predictions.

For evaluation, we use the standard COCO performance metrics: $AP_{50}$, $AP_{75}$, and AP (averaged over IoU thresholds), evaluated using mask IoU. Table 3 reports our instance segmentation results. We note that the performance of the Mask-RCNN framework increases consistently as we increase the temporal context for predictions. These results are visualized in Fig. 7.

| Method | $AP_{50}$ | $AP_{75}$ | AP |
|---|---|---|---|
| ResNet50-baseline | 0.610 | 0.305 | 0.321 |
| spatial G-CRF [6] | 0.618 | 0.310 | 0.329 |
| 2-frame st-G-CRF | 0.619 | 0.310 | 0.331 |
| 3-frame st-G-CRF | 0.631 | 0.321 | 0.330 |
| 4-frame st-G-CRF | 0.647 | 0.336 | 0.349 |

Table 3. Instance Segmentation using ResNet-50 Mask R-CNN on the Davis Person Dataset

| Method | mean IoU |
|---|---|
| Mask Track [30] | 79.7 |
| OSVOS [4] | 79.8 |
| Online Adaptation [38] | 85.6 |
| Online Adaptation + Spatial G-CRF [6] | 85.9 |
| Online Adaptation + 2-Frame st-G-CRF | 86.3 |
| Online Adaptation + 3-Frame st-G-CRF | 86.5 |

Table 4. Instance Tracking on the Davis val Dataset

### 3.2. Instance Tracking

Here we use the DAVIS dataset described in Sec. 3. Instance tracking involves predicting segmentation masks for foreground object instances for each frame in the video when presented with the ground truth segmentation for the first video frame. Our goal here is to demonstrate that incorporating temporal context helps improve performance in instance tracking methods. To this end, we extend the, state-of-the-art approach on the DAVIS benchmark, online adaptation approach from [38] with our spatio-temporal G-CRFs. We use their publicly available software based on the TensorFlow library to generate the unary terms for each of the frames in the video, and keep them fixed. We use a ResNet-50 network to generate spatio-temporal embeddings and use these alongside the unaries computed from [38]. The results are reported in table Table 4. We compare performance of spatio-temporal G-CRFs against that of just the unaries from [38], and also with spatial G-CRFs from [6]. The evaluation criterion is the mean pixel-IoU. It can be seen that temporal context helps improve the performance. We believe that re-implementing the software from [38] in Caffe2 and back-propagating on the unary branch of the network would yield further performance boosts.

| Model | Building | Tree | Sky | Car | Sign | Road | Pedestrian | Fence | Pole | Sidewalk | Cyclist | m-IoU |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SegNet [35] | 68.7 | 52.0 | 87.0 | 58.5 | 13.4 | 86.2 | 25.3 | 17.9 | 16.0 | 60.5 | 24.8 | 46.4 |
| Bayesian SegNet [1] | – | | | | | | | | | | | 63.1 |
| DeconvNet [16] | – | | | | | | | | | | | 48.9 |
| Visin et al. [37] | – | | | | | | | | | | | 58.8 |
| FCN8 [28] | 77.8 | 71.0 | 88.7 | 76.1 | 32.7 | 91.2 | 41.7 | 24.4 | 19.9 | 72.7 | 31.0 | 57.0 |
| DeepLab-LFOV [7] | 81.5 | 74.6 | 89.0 | 82.2 | 42.3 | 92.2 | 48.4 | 27.2 | 14.3 | 75.4 | 50.1 | 61.6 |
| Dilation8 [40] | 82.6 | 76.2 | 89.0 | 84.0 | 46.9 | 92.2 | 56.3 | 35.8 | 23.4 | 75.3 | 55.5 | 65.3 |
| Dilation8 + FSO [26] | 84.0 | 77.2 | 91.3 | 85.6 | 49.9 | 92.5 | 59.1 | 37.6 | 16.9 | 76.0 | 57.2 | 66.1 |
| Tiramisu [21] | 83.0 | 77.3 | 93.0 | 77.3 | 43.9 | 94.5 | 59.6 | 37.1 | 37.8 | 82.2 | 50.5 | 66.9 |
| Results with our ResNet-101 Implementation | | | | | | | | | | | | |
| Basenet ResNet-101 (Ours) | 81.2 | 75.1 | 90.3 | 85.2 | 48.3 | 93.9 | 57.7 | 39.9 | 15.9 | 80.5 | 54.8 | 65.7 |
| Basenet + Spatial G-CRF [6] | 81.6 | 75.7 | 90.4 | 86.8 | 48.1 | 94.0 | 59.1 | 39.2 | 15.7 | 80.7 | 54.7 | 66.0 |
| Basenet + 2-Frame Video G-CRF | 82.0 | 76.1 | 91.1 | 86.2 | 51.7 | 93.8 | 64.2 | 24.5 | 25.0 | 80.1 | 61.7 | 66.9 |
| Basenet + 3-Frame Video G-CRF | 82.1 | 76.0 | 91.1 | 86.1 | 52.0 | 93.7 | 64.5 | 24.9 | 24.4 | 79.9 | 61.8 | 67.0 |
| Results after Cityscapes Pretraining | | | | | | | | | | | | |
| Basenet ResNet-101 (Ours) | 85.5 | 77.4 | 90.9 | 88.4 | 62.3 | 95.4 | 64.8 | 62.1 | 33.3 | 85.5 | 60.5 | 73.3 |
| Basenet + Spatial G-CRF [6] | 86.0 | 77.8 | 91.2 | 90.8 | 63.6 | 95.9 | 66.5 | 61.2 | 35.3 | 86.9 | 65.8 | 74.6 |
| Basenet + 2-Frame Video G-CRF | 86.0 | 78.3 | 91.2 | 92.0 | 63.4 | 96.3 | 67.0 | 62.5 | 34.4 | 87.7 | 66.1 | 75.0 |
| Basenet + 3-Frame Video G-CRF | 86.1 | 78.3 | 91.2 | 92.2 | 63.7 | 96.4 | 67.3 | 63.0 | 34.4 | 87.8 | 66.4 | 75.2 |

Table 5. Results on CamVid dataset. We compare our results with some of the previously published methods, as well as our own implementation of the ResNet-101 network which serves as our base network.

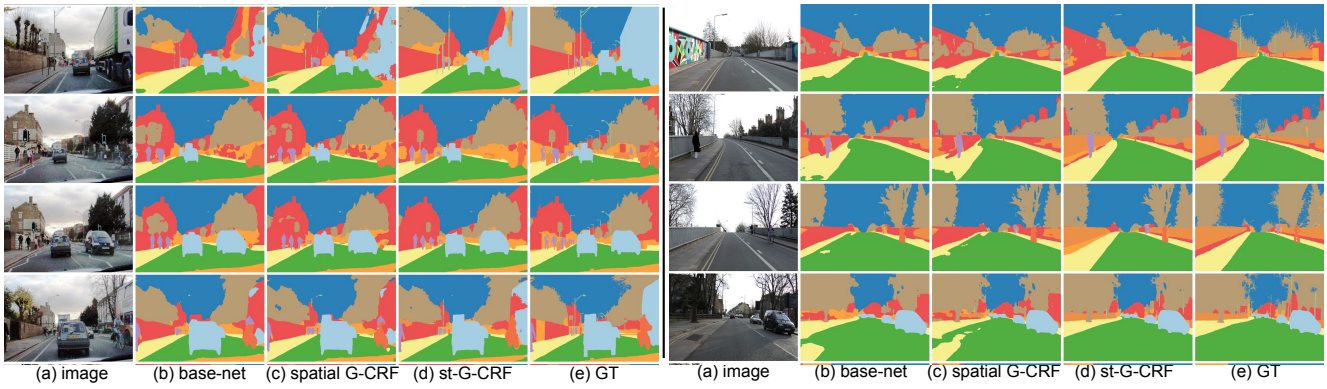



Figure 6. Qualitative results on the CamVid dataset. We note that the temporal context from neighbouring frames helps improve the prediction of the truck on the right in the first video, and helps distinguish between the road and the pavement in the second video, overall giving us smoother predictions in both cases.

### 3.3. Semantic Segmentation on CamVid Dataset

In this set of experiments, we employ our st-G-CRFs for the task of semantic video segmentation. Here we use the CamVid dataset described in Sec. 3. Our base network here is our own implementation of ResNet-101 with pyramid spatial pooling as in [41]. Additionally, we pretrain our networks on the Cityscapes dataset [11], and report results both with and without cityscapes pretraining. Without pretraining, we see an improvement of 1.3% over the base-net, and with pretraining we see an improvement of 1.9%. The qualitative results are shown in Fig. 6.

## 4. Conclusion

In this work, we propose efficient, end-to-end trainable G-CRFs for efficient spatio-temporal structured prediction. On a number of benchmarks, we experimentally demonstrate performance boosts when we increase the temporal context of predictions. This additional complexity comes at negligible computational overhead compared to spatial structured prediction owing to our efficient implementation. Future work would involve incorporating optical flow techniques in this framework as they provide a natural way of capturing temporal correspondence. Further, we would like to employ our method for spatio-temporal structured prediction in unsupervised and semi-supervised settings.


Figure 7. Qualitative results for instance segmentation on the DAVIS Person Dataset. We notice that the base-net and the spatial G-CRF in (a),(b) miss instances or parts of instances of dancing persons frequently. Temporal context from spatio-temporal G-CRFs in (c) helps recover missing parts / instances and yield smoother predictions over the video, as seen in row 2.

# Appendix

## A. Gradient Expressions for Spatio-Temporal G-CRF Parameters

As described in the manuscript, to capture the spatio-temporal context, we propose two kinds of pairwise interactions: (a) pairwise terms between patches in the same frame (spatial pairwise terms), and (b) pairwise terms between patches in different frames (temporal pairwise terms).

Denoting the spatial pairwise terms at frame $v$ by $A_v$ and the temporal pairwise terms between frames $u, v$ as $T_{u,v}$, our inference equation is written as

$$\begin{bmatrix} A_1 + \lambda \mathbf{I} & T_{1,2} & \cdots & T_{1,V} \\ T_{2,1} & A_2 + \lambda \mathbf{I} & \cdots & T_{2,V} \\ & & \vdots & \\ T_{V,1} & T_{V,2} & \cdots & A_V + \lambda \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_V \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_V \end{bmatrix}, \tag{A.1}$$

where we group the variables by frames. Solving this system allows us to couple predictions $\mathbf{x}_v$ across all video frames $v \in \{1, \dots, V\}$, positions, $p$ and labels $l$. If furthermore $A_v = A'_v, \forall v$ and $T_{u,v} = T'_{v,u}, \forall u, v$ then the resulting system is positive definite for any positive $\lambda$.

As in the manuscript, at frame $v$ we couple the scores for a pair of patches $p_i, p_j$ taking the labels $l_m, l_n$ respectively as follows:

$$A_{v,p_i,p_j}(l_m, l_n) = \langle \mathcal{A}^{l_m}_{v,p_i}, \mathcal{A}^{l_n}_{v,p_j} \rangle, \tag{A.2}$$

where $i, j \in \{1, \dots, P\}$ and $m, n \in \{1, \dots, L\}$, $v \in \{1, \dots, V\}$, and $\mathcal{A}^{l_n}_{v,p_j} \in R^D$ is the embedding associated to point $p_j$.

Thus, $\mathcal{A}_v \in \mathbb{R}^{N \times D}$, where $N = P \times L$. Further, to design the *temporal* pairwise terms, we couple patches $p_i, p_j$ coming from different frames $u, v$ taking the labels $l_m, l_n$ respectively as

$$T_{u,v,p_i,p_j}(l_m, l_n) = \langle \mathcal{T}^{l_m}_{u,p_i}, \mathcal{T}^{l_n}_{v,p_j} \rangle, \tag{A.3}$$

where $u, v \in \{1, \dots, V\}$.

In short, both the spatial pairwise and the temporal pairwise terms are composed as Gram matrices of spatial and temporal embeddings as $A_v = \mathcal{A}_v^\top \mathcal{A}_v$, and $T_{u,v} = \mathcal{T}_u^\top \mathcal{T}_v$.

Using the definitions from Eq. A.2 and Eq. A.3, we can rewrite the inference equation as

$$\begin{bmatrix} \mathcal{A}_1^T\mathcal{A}_1 + \lambda\mathbf{I} & \mathcal{T}_1^T\mathcal{T}_2 & \cdots & \mathcal{T}_1^T\mathcal{T}_V \\ \mathcal{T}_2^T\mathcal{T}_1 & \mathcal{A}_2^T\mathcal{A}_2 + \lambda\mathbf{I} & \cdots & \mathcal{T}_2^T\mathcal{T}_V \\ & & \vdots & \\ \mathcal{T}_V^T\mathcal{T}_1 & \mathcal{T}_V^T\mathcal{T}_2 & \cdots & \mathcal{A}_V^T\mathcal{A}_V + \lambda\mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_V \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_V \end{bmatrix} \tag{A.4}$$

From Eq. A.4, we can express $\mathbf{b}_v$ as follows:

$$\mathbf{b}_v = \mathcal{A}_v^T\mathcal{A}_v\mathbf{x}_v + \lambda\mathbf{x}_v + \sum_{u \neq v} \mathcal{T}_v^T\mathcal{T}_u\mathbf{x}_u, \tag{A.5}$$

which can be compactly written as

$$\mathbf{b}_v = A_v\mathbf{x}_v + \lambda\mathbf{x}_v + \sum_{u \neq v} T_{v,u}\mathbf{x}_u. \tag{A.6}$$

We will use Eq. A.6 to derive gradient expressions for $\frac{\partial \mathcal{A}_v}{\partial \mathcal{L}}$ and $\frac{\partial \mathcal{T}_v}{\partial \mathcal{L}}$.

## A.1. Gradients of the Unary Terms

As in [5, 6], the gradients of the unary terms $\frac{\partial \mathbf{b}_v}{\partial \mathcal{L}}$ are obtained from the solution of the following system of linear equations:

$$\begin{bmatrix} \mathcal{A}_1^T\mathcal{A}_1 + \lambda\mathbf{I} & \mathcal{T}_1^T\mathcal{T}_2 & \cdots & \mathcal{T}_1^T\mathcal{T}_V \\ \mathcal{T}_2^T\mathcal{T}_1 & \mathcal{A}_2^T\mathcal{A}_2 + \lambda\mathbf{I} & \cdots & \mathcal{T}_2^T\mathcal{T}_V \\ & & \vdots & \\ \mathcal{T}_V^T\mathcal{T}_1 & \mathcal{T}_V^T\mathcal{T}_2 & \cdots & \mathcal{A}_V^T\mathcal{A}_V + \lambda\mathbf{I} \end{bmatrix} \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \mathbf{b}_1} \\ \frac{\partial \mathcal{L}}{\partial \mathbf{b}_2} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \mathbf{b}_V} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial \mathbf{x}_1} \\ \frac{\partial \mathcal{L}}{\partial \mathbf{x}_2} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial \mathbf{x}_V} \end{bmatrix}, \tag{A.7}$$

where $\mathcal{L}$ is the network loss. Once we have $\frac{\partial \mathcal{L}}{\partial \mathbf{b}_v}$, we use it to compute the gradients of the spatio-temporal embeddings.

## A.2. Gradients of the Spatial Embeddings

We begin with the observation that computing $\frac{\partial \mathcal{L}}{\partial \mathcal{A}_v}$ requires us to first derive the expression for $\frac{\partial \mathcal{L}}{\partial A_v}$. To this end, we ignore terms from Eq. A.6 that do not depend on $\mathbf{b}_v$ or $A_v$ and write it as $\mathbf{b}_v = A_v\mathbf{x}_v + c$. We now use the result from [5, 6] that when

$$A_v\mathbf{x}_v = \mathbf{b}_v,$$

the gradients of $A_v$ are expressed as

$$\frac{\partial \mathcal{L}}{\partial A_v} = -\frac{\partial \mathcal{L}}{\partial \mathbf{b}_v} \otimes \mathbf{x}_v, \tag{A.8}$$

where $\otimes$ denotes the Kronecker product operator.

To compute $\frac{\partial \mathcal{L}}{\partial \mathcal{A}_v}$, we use the chain rule of differentiation as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{A}_v} = \left(\frac{\partial \mathcal{L}}{\partial A_v}\right)\left(\frac{\partial A_v}{\partial \mathcal{A}_v}\right) = \left(\frac{\partial \mathcal{L}}{\partial A_v}\right)\left(\frac{\partial}{\partial \mathcal{A}_v}\mathcal{A}_v^T\mathcal{A}_v\right), \tag{A.9}$$

where $A_v = \mathcal{A}_v^T\mathcal{A}_v$, by definition. We know the expression for $\frac{\partial \mathcal{L}}{\partial A_v}$ from Eq. A.8, but to obtain the expression for $\frac{\partial}{\partial \mathcal{A}_v}\mathcal{A}_v^T\mathcal{A}_v$ we define a permutation matrix $Q_{m,n}$ of size $mn \times mn$ (as in [13, 6]) as follows:

$$Q_{m,n}\text{vec}(M) = \text{vec}(M^T), \tag{A.10}$$

where $\text{vec}(M)$ is the vectorization operator that vectorizes a matrix $M$ by stacking its columns. Thus, the operator $Q_{m,n}$ is a permutation matrix, composed of 0s and 1s, and has a single 1 in each row and column. When premultiplied with another matrix, $Q_{m,n}$ rearranges the ordering of rows of that matrix, while when postmultiplied with another matrix, $Q_{m,n}$ rearranges its columns. Using this matrix, we can form the following expression [13]:

$$\frac{\partial}{\partial \mathcal{A}_v}\mathcal{A}_v^T\mathcal{A}_v = \left(\mathbf{I} \otimes \mathcal{A}_v^T\right) + \left(\mathcal{A}_v^T \otimes \mathbf{I}\right) Q_{D,N}, \tag{A.11}$$

where $\mathbf{I}$ is the $N \times N$ identity matrix. Substituting Eq. A.8 and Eq. A.11 into Eq. A.9, we obtain:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{A}_v} = -\left(\frac{\partial \mathcal{L}}{\partial \mathbf{b}_v} \otimes \mathbf{x}_v\right)\left(\left(\mathbf{I} \otimes \mathcal{A}_v^\top\right) + \left(\mathcal{A}_v^\top \otimes \mathbf{I}\right) Q_{D,N}\right). \tag{A.12}$$

## A.3. Gradients of Temporal Embeddings

As in the last section, from Eq. A.6, we ignore any terms that do not depend on $\mathbf{b}_v$ or $T_{v,u}$ and write it as $\mathbf{b}_v = c + \sum_{u \neq v} T_{v,u}\mathbf{x}_u$.

Using the strategies in the previous section and the sum rule of differentiation, the gradients of the temporal embeddings are given by the following form:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{T}_v} = -\sum_u \left(\frac{\partial \mathcal{L}}{\partial \mathbf{b}_u} \otimes \mathbf{x}_v\right)\left(\left(\mathbf{I} \otimes \mathcal{T}_u^\top\right) + \left(\mathcal{T}_u^\top \otimes \mathbf{I}\right) Q_{D,N}\right) \tag{A.13}$$

## References


[1] V. B. A. Kendall and R. Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. In *ArXiV* CoRR, *abs/1511.02680*, 2015. 8

[2] Y. Adi, J. Keshet, E. Cibelli, and M. Goldrick. Sequence segmentation using joint RNN and structured prediction models. In *ICASSP*, pages 2422–2426, 2017. 2

[3] S. Bratieres, N. Quadrianto, and Z. Ghahramani. GPstruct: Bayesian structured prediction using Gaussian processes. *IEEE trans. on pattern analysis and machine intelligence*, 37(7):1514–1520, 2015. 2

[4] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool. One-shot video object segmentation. In *CVPR*, 2017. 2, 7

[5] S. Chandra and I. Kokkinos. Fast, exact and multi-scale inference for semantic image segmentation with deep Gaussian CRFs. In *ECCV*, 2016. 1, 2, 3, 10

[6] S. Chandra and I. Kokkinos. Dense and low-rank Gaussian CRFs using deep embeddings. In *ICCV*, 2017. 1, 2, 3, 4, 5, 7, 8, 10

[7] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. *ICLR*, 2015. 2, 8

[8] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *arXiv:1606.00915*, 2016. 6

[9] L.-C. Chen, G. Papandreou, K. Murphy, and A. L. Yuille. Weakly- and semi-supervised learning of a deep convolutional network for semantic image segmentation. *ICCV*, 2015. 2

[10] L.-C. Chen, A. G. Schwing, A. L. Yuille, and R. Urtasun. Learning Deep Structured Models. In *ICML*, 2015. 2

[11] M. Cordts, M. Omran, S. Ramos, T. Scharwachter, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. *CVPR*, 2016. 8

[12] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, pages 2625–2634, 2015. 2

[13] P. L. Fackler. Notes on matrix calculus. 2005. 10

[14] J. F. G. J. Brostow, J. Shotton and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV*, 2017. 5

[15] R. Gadde, V. Jampani, and P. V. Gehler. Semantic video CNNs through representation warping. In *ICCV*, 2017. 2

[16] S. H. H. Noh and B. Han. Learning deconvolution network for semantic segmentation. In arXiv preprint arXiv:1505.04366, 2015. 8

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. *ICCV*, 2017. 7

[18] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *CVPR*, Jul 2017. 2

[19] S. Jain, B. Xiong, and K. Grauman. Fusionseg: Learning to combine motion and appearance for fully automatic segmention of generic objects in videos. *arXiv preprint arXiv:1701.05384*, 2017. 2

[20] V. Jampani, R. Gadde, and P. V. Gehler. Video propagation networks. In *CVPR*, 2017. 2

[21] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pages 1175–1183. IEEE, 2017. 5, 8

[22] X. Jin, X. Li, H. Xiao, X. Shen, Z. Lin, J. Yang, Y. Chen, J. Dong, L. Liu, Z. Jie, J. Feng, and S. Yan. Video scene parsing with predictive feature learning. *CoRR*, abs/1612.00119, 2016. 2

[23] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, pages 1725–1732, 2014. 2

[24] A. Khoreva, R. Benenson, E. Ilg, T. Brox, and B. Schiele. Lucid data dreaming for object tracking. *arXiv preprint arXiv:1703.09554*, 2017. 2

[25] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. In *NIPS*, 2011. 1, 2, 5

[26] A. Kundu, V. Vineet, and V. Koltun. Feature space optimization for semantic video segmentation. In *CVPR*, pages 3168–3175, 2016. 1, 2, 5, 8

[27] X. Li, Y. Qi, Z. Wang, K. Chen, Z. Liu, J. Shi, P. Luo, X. Tang, and C. C. Loy. Video object segmentation with re-identification. *CVPR workshops - The 2017 DAVIS Challenge on Video Object segmentation*, 2017. 2

[28] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 8

[29] D. Nilsson and C. Sminchisescu. Semantic video segmentation by gated recurrent flow propagation. *CoRR*, abs/1612.08871, 2016. 2

[30] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A.Sorkine-Hornung. Learning video object segmentation from static images. In *Computer Vision and Pattern Recognition*, 2017. 2, 7

[31] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *CVPR*, 2016. 5

[32] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017. 5

[33] E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell. Clockwork convnets for video semantic segmentation. *CoRR*, abs/1608.03609, 2016. 2

[34] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. In

*https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf*, 1994. 3, 4

[35] A. K. V. Badrinarayanan and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. In *ArXiV* CoRR*, abs/1511.00561*, 2015. 5, 8

[36] R. Vemulapalli, O. Tuzel, M.-Y. Liu, and R. Chellapa. Gaussian conditional random field network for semantic segmentation. In *CVPR*, June 2016. 2

[37] F. Visin, M. Ciccone, A. Romero, K. Kastner, K. Cho, Y. Bengio, M. Matteucci, and A. Courville. Reseg: A recurrent neural network-based model for semantic segmentation. In *CVPR workshop*, 2016. 8

[38] P. Voigtlaender and B. Leibe. Online adaptation of convolutional neural networks for video object segmentation. In *BMVC*, 2017. 2, 5, 7

[39] T.-H. Vu, A. Osokin, and I. Laptev. Context-aware CNNs for person head detection. In *ICCV*, pages 2893–2901, 2015. 2

[40] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *ICLR*, 2016. 8

[41] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *CoRR*, abs/1612.01105, 2016. 2, 8

[42] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015. 2, 4

[43] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei. Deep feature flow for video recognition. *CoRR*, abs/1611.07715, 2016. 2