Deep Spatio-Temporal Random Fields for Efficient Video Segmentation

Siddhartha Chandra[1] Camille Couprie[2] Iasonas Kokkinos[2]
siddhartha.chandra@inria.fr coupriec@fb.com 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 Conditional Random Fields (GCRFs). Our method, called VideoGCRF is (a) efficient, (b) has a unique global minimum, and (c) can be trained end-to-end alongside contemporary deep networks for video understanding. 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 implementation is based on the Caffe2 *framework and will be available at* https://github.com/siddharthachandra/gcrf-v3.0.

1. Introduction

Video understanding remains largely unsolved despite significant improvements in image understanding over the past few years. The accuracy of current image classification and 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 the time variable. Combined with the increase in memory and computation demands, video understanding poses additional challenges that call for novel methods.

Our objective in this work is to couple the decisions taken by a neural network in time, in a manner that allows information to flow across frames and thereby result 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 VideoGCRF, an extension into video segmentation of the Deep Gaussian Random Field (DGRF) technique recently proposed for single-frame structured prediction in [5, 6].

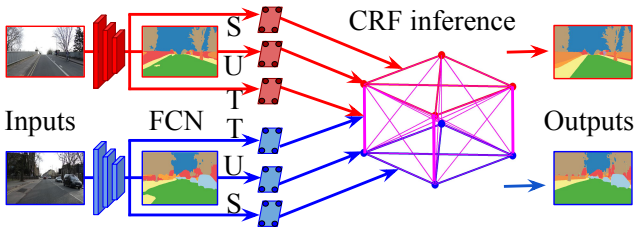

Figure 1: Overview of our VideoGCRF approach: we jointly segment multiple images by passing them firstly through a fully convolutional network to obtain per-pixel class scores ('unary' terms U), alongside with spatial (S) and temporal (T) embeddings. We couple predictions at different spatial and temporal positions in terms of the inner product of their respective embeddings, shown here as arrows pointing to a graph edge. The final prediction is obtained by solving a linear system; this can eliminate spurious responses, e.g. on the left pavement, by diffusing the per-pixel node scores over the whole spatio-temporal graph. The CRF and CNN architecture is jointly trained end-to-end, while CRF inference is exact and particularly efficient.

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 [24, 25], (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 [41] (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 [24, 25], and (d) 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 [25] on Feature Space Optimization. Even though our works share several conceptual similarities, our method is entirely different at the technical level. In our case spatio-temporal inference is implemented as a structured, 'lateral connection' layer that is trained jointly with the feed-forward CNNs, while the method of [25] 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, 35, 38, 41] 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 capture spatio-temporal context involved designing deep learning architectures [22] that implicitly learn interactions between consecutive image frames. A number of subsequent approaches used Recurrent Neural Networks (RNNs) [2, 12] to capture inter-dependencies between the image frames. Other approaches have exploited optical flow computed from state of the art approaches [17] as additional input to the network [14, 18]. Finally, [25] explicitly capture temporal constraints via pairwise terms over probabilistic graphical models, but operate post-hoc, i.e. are not trained jointly with the underlying network.

In this work, we focus on three problems, namely (i) semantic and (ii) instance video segmentation as well as (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 video frames. The first set of approaches to address this task 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, 37], optionally leveraging a variety of data augmentation regimes [23] 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 [29], where the goal is to warp the segmentation of the first frame using the images and optical flow as additional inputs [19, 23, 26].

A number of approaches have attempted to exploit temporal information to improve over static image segmentation approaches for video segmentation. Clockwork convnets [32] 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 [25, 42]. In [28] segmentations are warped using the flow and spatial transformer networks. Rather than using optical flow, the prediction of future segmentations [21] may also temporally smooth results obtained frame-by-frame. Finally, the state-of-the-art on this task [14] improves over PSPnet[40] by warping the feature maps of a static segmentation CNN to emulate a video segmentation network.

2. VideoGCRF

In this work we introduce VideoGCRF, extending 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 to integrate information over time, allowing us 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 of scores for 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.

The Gaussian-CRF (or, G-CRF) model defines a joint posterior distribution through a Gaussian multivariate density for a video as:

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

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 we assume that A, B depend on the input video and we omit the conditioning on $\mathcal{V}$ for convenience.

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

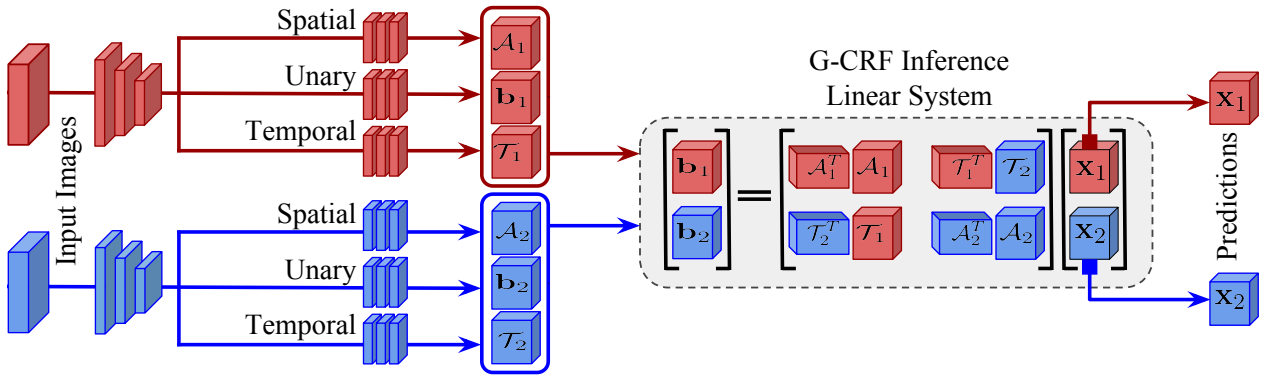

Figure 2: VideoGCRF schematic for 2 video frames. Our network takes in 2 input images, and delivers the per frame unaries $\mathbf{b}_1, \mathbf{b}_2$, spatial embeddings $\mathcal{A}_1, \mathcal{A}_2$, and temporal embeddings $\mathcal{T}_1, \mathcal{T}_2$ in the feed-forward mode. Our VideoGCRF module collects these and solves the inference problem in Eq. 2 to recover predictions $\mathbf{x}_1, \mathbf{x}_2$. During backward pass, the gradients of the predictions are delivered to the VideoGCRF 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.

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 λ 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 [33] 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 [24], 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 λ.

We now describe how the pairwise terms $A_v, T_{u,v}$ are constructed through our CNN, and then discuss acceleration of the linear system in Eq. 2 by exploiting its structure.

Spatial Connections: We define the spatial pairwise terms in terms of inner products of pixel-wise embeddings, as in [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

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

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.

VideoGCRF in Deep Learning: Our proposed spatio-temporal Gaussian CRF (VideoGCRF) 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 [33], described in Algorithm 1 that is customized for our VideoGCRFs.

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, which we eliminate 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

Since we rely 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. Thanks to this we do not have to perform back-propagation in time which was needed e.g. in [41] 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, as in [6].

2.4. Implementation and Inference Time

Our implementation is GPU based and exploits fast *CUDA-BLAS* linear algebra routines. It is implemented as a module in the Caffe2 library. For spatial and temporal embeddings of size 128, 12 classes (Sec. 3.3), a 321×321 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 DenseCRF method [24] which takes 0.2s on average for spatial CRF for a single input frame. 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. 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 [37] 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).

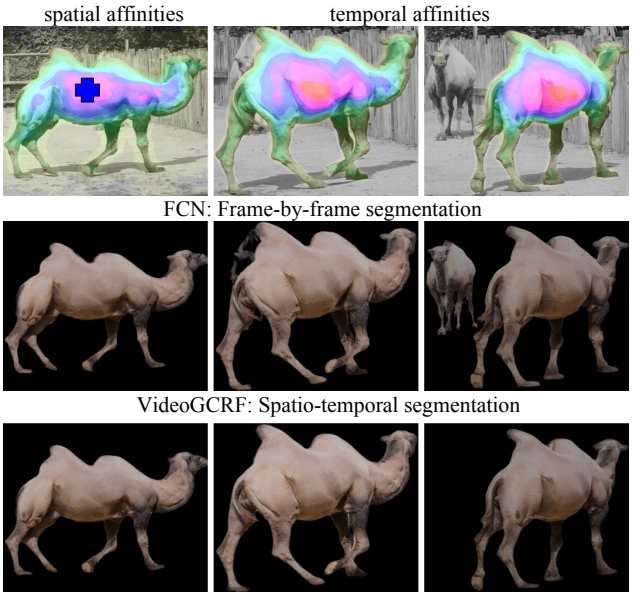

Figure 3: Visualization of instance segmentation through VideoGCRF: In row 1 we focus on a single point of the CRF graph, shown as a cross, and show as a heatmap its spatial (inter-frame) and temporal (intra-frame) affinities to all other graph nodes. These correspond to a single column of the linear system in Eq. 2. In row 2 we show the predictions that would be obtained by frame-by-frame segmentation, relying exclusively on the FCN's unary terms, while in row 3 we show the results obtained after solving the VideoGCRF inference problem. We observe that in frame-by-frame segmentation a second camel is incorrectly detected due to its similar appearance properties. However, VideoGCRF inference exploits temporal context and focuses solely on the correct object.

Datasets. We use the three datasets for our experiments:

DAVIS. The DAVIS dataset [30] 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 [31] provides densely annotated frames for instance segmentation, it lacks object category labels. For category prediction tasks such

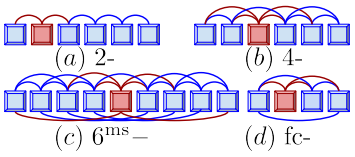

Figure 4: Temporal neighbourhoods in our ablation study: boxes denote video frames and the arcs connecting them are pairwise connections. The frame in red has all neighbours present in the temporal context.

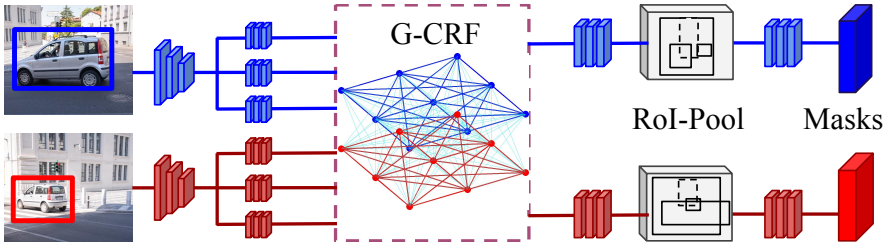

Figure 5: Spatio-temporal structured prediction in Mask-RCNN. Here we use CRFs in the feature learning stage before the ROI-Pooling (and not as the final classifier). This helps learn mid-level features which are better aware of the spatio-temporal context.

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 [13, 3], 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 [34, 25, 20], 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 VideoGCRF. For these set of experiments, our baseline network, or *base-net* is a single resolution ResNet-101 network, with altered network strides as in [8] to produce a spatial down-sampling factor of 8. The evaluation metric used 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 for $2-$frame inference. Our best results are achieved at spatio-temporal embeddings of size 128. The improvement over the base-net is 4.2%. In subsequent experiments 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 GPU RAM: for a ResNet-101 network, an input image of size 321×321, embeddings of size 128, we can currently fit $\mathcal{V} = 7$ frames on 12 GB of GPU RAM. 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.

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 combinations 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 a moderate decrease in performance. This could be a consequence of the long-range interactions having the same weight as short-range interactions. In the future we intend to mitigate this issue by complementing our embeddings with the temporal distance between frames.

Instance Segmentation Experiments. We now demonstrate the utility of our VideoGCRF method for the task of proposal-based instance segmentation. Our hypothesis is that coupling predictions across frames is advantageous for instance segmentation methods. We actually show that the performance of the instance segmentation methods improves as we increase the temporal context via VideoGCRF, and obtain our best results with fully-connected temporal neighbourhoods. Our baseline for this task is the Mask-RCNN framework of [16] using the ResNet-50 network as the convolutional *body*. The Mask-RCNN framework uses

base-net	**81.16**			
VideoGCRF	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. We study the effect of varying the size of the spatial & temporal embeddings.

base-net	**81.16**			
VideoGCRF	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.

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 VideoGCRF before the RoI pooling stage as shown in Fig. 5. Instead of coupling final predictions, we thereby couple mid-level features over the video frames, thereby improving the features which are ultimately 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.

3.2. Instance Tracking

We use the DAVIS dataset described in Sec. 3. Instance tracking involves predicting foreground segmentation masks for each video frame given the foreground segmentation for the first video frame. We demonstrate that incorporating temporal context helps improve performance in instance tracking methods. To this end we extend the online adaptation approach of [37] which is the state-of-the-art approach on the DAVIS benchmark with our VideoGCRF. 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 [37]. The results are reported in table Table 4. We compare performance of VideoGCRF against that of just the unaries from [37], and also with spatial CRFs from [6]. The evaluation criterion is the mean pixel-IoU. It can be seen that temporal context improves performance. We hypothesize that re-implementing the software from [37] in Caffe2 and back-propagating on the unary branch of the network would yield further improvements.

Method	AP_{50}	AP_{75}	AP
ResNet50-baseline	0.610	0.305	0.321
spatial CRF [6]	0.618	0.310	0.329
2-frame VideoGCRF	0.619	0.310	0.331
3-frame VideoGCRF	0.631	0.321	0.330
4-frame VideoGCRF	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 [29]	79.7
OSVOS [4]	79.8
Online Adaptation [37]	85.6
Online Adaptation + Spatial CRF [6]	85.9
Online Adaptation + 2-Frame VideoGCRF	86.3
Online Adaptation + 3-Frame VideoGCRF	86.5

Table 4: Instance Tracking on the Davis val Dataset

3.3. Semantic Segmentation on CamVid Dataset

We now employ our VideoGCRF for the task of semantic video segmentation on the CamVid dataset. Our base network here is our own implementation of ResNet-101 with pyramid spatial pooling as in [40]. Additionally, we pretrain our networks on the Cityscapes dataset [11], and report results both with and without pretraining on Cityscapes. We report improvements over the baseline networks in both settings. 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. We notice that VideoGCRF benefits from temporal context, yielding smoother predictions across video frames.

Model	Building	Tree	Sky	Car	Sign	Road	Pedestrian	Fence	Pole	Sidewalk	Cyclist	m-IoU
DeconvNet [15]						–						48.9
SegNet [34]	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
Visin et al. [36]						–						58.8
FCN8 [27]	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 [39]	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 [25]	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 [20]	83.0	77.3	93.0	77.3	43.9	94.5	59.6	37.1	37.8	82.2	50.5	66.9
Gadde et al. [14]						–						67.1
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 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 VideoGCRF	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 VideoGCRF	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 + denseCRF post-processing [24]	84.3	76.1	90.5	88.9	65.1	95.4	65.4	61.5	34.1	85.8	66.2	73.9
Basenet + Spatial 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 VideoGCRF	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 VideoGCRF	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.

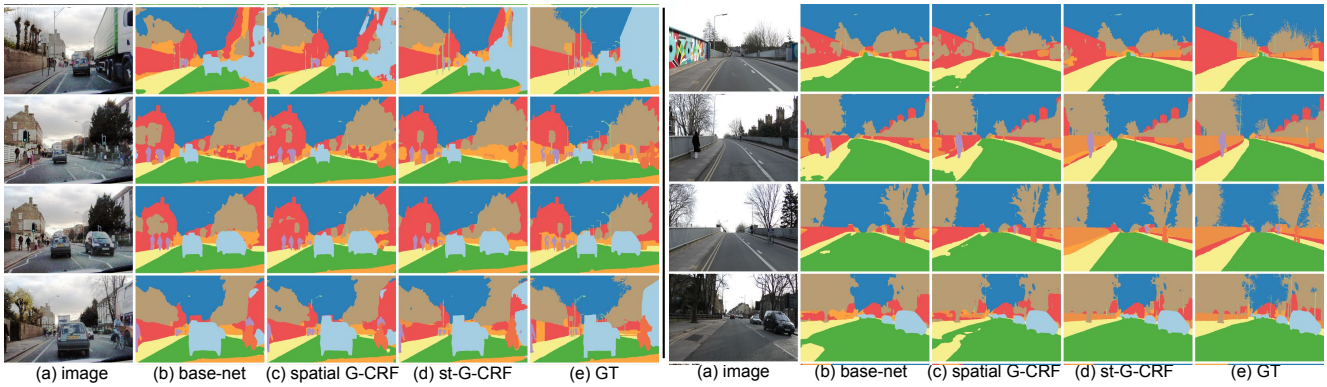

Figure 6: Qualitative results on the CamVid dataset. We note that the temporal context from neighbouring frames helps improve the prediction of the truck on the right in the first video, and helps distinguish between the road and the pavement in the second video, overall giving us smoother predictions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

4. Conclusion

In this work, we propose VideoGCRF, an end-to-end trainable Gaussian CRF for efficient spatio-temporal structured prediction. We empirically show performance improvements on several benchmarks thanks to an increase of the temporal context. This additional functionality comes at negligible computational overhead owing to efficient implementation. In future work we want to incorporate optical flow techniques to capture temporal correspondence and use temporal distance between frames to complement our embeddings. Finally, we believe that our method for spatio-temporal structured prediction can prove useful in the unsupervised and semi-supervised setting.

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] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 2008. 6

[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

[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

[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. 7

[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] J. F. G. J. Brostow, J. Shotton and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV*, 2017. 6

[14] R. Gadde, V. Jampani, and P. V. Gehler. Semantic video CNNs through representation warping. In *ICCV*, 2017. 2, 8

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

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. *ICCV*, 2017. 6

[17] 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

[18] 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

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

[20] 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. 6, 8

[21] 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

[22] 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

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

[24] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. In *NIPS*, 2011. 1, 3, 5, 8

[25] A. Kundu, V. Vineet, and V. Koltun. Feature space optimization for semantic video segmentation. In *CVPR*, pages 3168–3175, 2016. 1, 2, 6, 8

[26] 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

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

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

[29] 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

[30] 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

[31] 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

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

[33] 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

[34] A. K. V. Badrinarayanan and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. In *ArXiV* CoRR, *abs/1511.00561*, 2015. 6, 8

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

[36] 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

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

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

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

[40] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *CoRR*, abs/1612.01105, 2016. 2, 7

[41] 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. 1, 2, 4

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