Recurrent Temporal Deep Field for Semantic Video Labeling

Peng Lei and Sinisa Todorovic

School of Electrical Engineering and Computer Science
Oregon State University
leip@oregonstate.edu, sinisa@eecs.oregonstate.edu

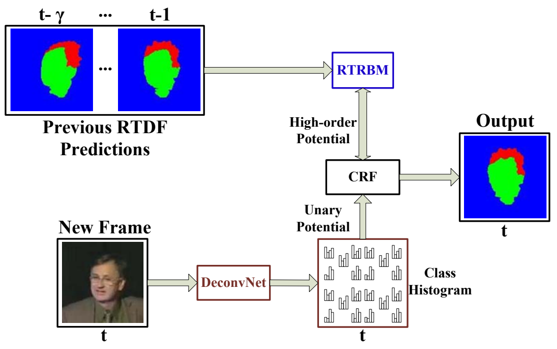
Abstract. This paper specifies a new deep architecture, called Recurrent Temporal Deep Field (RTDF), for semantic video labeling. RTDF is a conditional random field (CRF) that combines a deconvolution neural network (DeconvNet) and a recurrent temporal restricted Boltzmann machine (RTRBM). DeconvNet is grounded onto pixels of a new frame for estimating the unary potential of the CRF. RTRBM estimates a high-order potential of the CRF by capturing long-term spatiotemporal dependencies of pixel labels that RTDF has already predicted in previous frames. We derive a mean-field inference algorithm to jointly predict all latent variables in both RTRBM and CRF. We also conduct end-to-end joint training of all DeconvNet, RTRBM, and CRF parameters. The joint learning and inference integrate the three components into a unified deep model – RTDF. Our evaluation on the benchmark Youtube Face Database (YFDB) and Cambridge-driving Labeled Video Database (Camvid) demonstrates that RTDF outperforms the state of the art both qualitatively and quantitatively.

Keywords: Video Labeling, Recurrent Temporal Deep Field, Recurrent Temporal Restricted Boltzmann Machine, Deconvolution, CRF

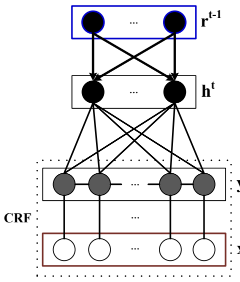
1 Introduction

This paper presents a new deep architecture for semantic video labeling, where the goal is to assign an object class label to every pixel. Our videos show natural driving scenes, captured by a camera installed on a moving car facing forward, or indoor close-ups of a person's head facing the camera. Both outdoor and indoor videos are recorded in uncontrolled environments with large variations in lighting conditions and camera viewpoints. Also, objects occurring in these videos exhibit a wide variability in appearance, shape, and motion patterns, and are subject to long-term occlusions. To address these challenges, our key idea is to efficiently account for both local and long-range spatiotemporal cues using deep learning.

Our deep architecture, called Recurrent Temporal Deep Field (RTDF), leverages the conditional random field (CRF) [4] for integrating local and contextual visual cues toward semantic pixel labeling, as illustrated in Fig. 1. The energy of RTDF is defined in terms of unary, pairwise, and higher-order potentials.



(a)



(b)

Fig. 1. (a) Our semantic labeling for a Youtube Face video [1] using RTDF. Given a frame at time t, RTDF uses a CRF to fuse both local and long-range spatiotemporal cues for labeling pixels in frame t. The local cues (red box) are extracted by DeconvNet [2] using only pixels of frame t. The long-range spatiotemporal cues (blue box) are estimated by RTRBM [3] (precisely, the hidden layer of RTDF) using a sequence of previous RTDF predictions for pixels in frames $t{-}1, t{-}2, \ldots, t{-}\gamma$. (b) An illustration of RTDF with pixel labels $\mathbf{y}^t$ in frame t, unary potentials $\mathbf{x}^t$, and top two layers $\mathbf{r}^{t-1}$ and $\mathbf{h}^t$ belonging to RTRBM. The high-order potential is distributed to all pixels in frame t via the full connectivity of layers $\mathbf{h}^t$ and $\mathbf{y}^t$, and layers $\mathbf{r}^{t-1}$ and $\mathbf{h}^t$.

As the unary potential, we use class predictions of the Deconvolution Neural Network (DeconvNet) [2] for every pixel of a new frame at time t. DeconvNet efficiently computes the unary potential in a feed-forward manner, through a sequence of convolutional and deconvolutional processing of pixels in frame t. Since the unary potential is computed based only on a single video frame, DeconvNet can be viewed as providing local spatial cues to our RTDF. As the pairwise potential, we use the standard spatial smoothness of pixel labels. Finally, as the higher-order potential, we use hidden variables of the Recurrent Temporal Restricted Boltzmann Machine (RTRBM) [3] (see Fig. 1b). This hidden layer of RTRBM is computed from a sequence of previous RTDF predictions for pixels in frames $\{t{-}1, t{-}2, \ldots, t{-}\gamma\}$. RTRBM is aimed at capturing long-range spatiotemporal dependencies among already predicted pixel labels, which is then used to enforce spatiotemporal coherency of pixel labeling in frame t.

We formulate a new mean-field inference algorithm to jointly predict all latent variables in both RTRBM and CRF. We also specify a joint end-to-end learning of CRF, DeconvNet and RTRBM. The joint learning and inference integrate the three components into a unified deep model – RTDF.

The goal of inference is to minimize RTDF energy. Input to RTDF inference at frame t consists of: (a) pixels of frame t, and (b) RTDF predictions for pixels in frames $\{t-1,\ldots,t-\gamma\}$. Given this input, our mean-field inference algorithm *jointly* predicts hidden variables of RTRBM and pixel labels in frame t.

Parameters of CRF, DeconvNet, and RTRBM are *jointly* learned in an end-to-end fashion, which improves our performance over the case when each component of RTDF is independently trained (a.k.a. piece-wise trained).

Our semantic video labeling proceeds frame-by-frame until all frames are labeled. Note that for a few initial frames $t \leq \gamma$, we do not use the high-order potential, but only the unary and pairwise potentials in RTDF inference.

Our contributions are summarized as follows:

1. A new deep architecture, RTDF, capable of efficiently capturing both local and long-range spatiotemporal cues for pixel labeling in video,
2. An efficient mean-field inference algorithm that jointly predicts hidden variables in RTRBM and CRF and labels pixels; as our experiments demonstrate, our mean-field inference yields better accuracy of pixel labeling than an alternative stage-wise inference of each component of RTDF.
3. A new end-to-end joint training of all components of RTDF using loss back-propagation; as our experiments demonstrate, our joint training outperforms the case when each component of RTDF is trained separately.
4. Improved pixel labeling accuracy relative to the state of the art, under comparable runtimes, on the benchmark datasets.

In the following, Sec. 2 reviews closely related work; Sec. 3 specifies RTDF and briefly reviews its basic components: RBM in Sec. 3.1, RTRBM in Sec. 3.2, and DeconvNet in Sec. 3.3; Sec. 4 formulates RTDF inference; Sec. 5 presents our training of RTDF; and Sec. 6 shows our experimental results.

2 Related Work

This section reviews closely related work on semantic video labeling, whereas the literature on unsupervised and semi-supervised video segmentation is beyond our scope. We also discuss our relationship to other related work on semantic image segmentation, and object shape modeling.

Semantic video labeling has been traditionally addressed using hierarchical graphical models (e.g., [5–10]). However, they typically resort to extracting hand-designed video features for capturing context, and compute compatibility terms only over local space-time neighborhoods.

Our RTDF is related to semantic image segmentation using CNNs [12–20]. These approaches typically use multiple stages of training, or iterative component-wise training. Instead, we use a joint training of all components of our deep architecture. For example, a fully convolutional network (FCN) [13] is trained in a stage-wise manner such that a new convolution layer is progressively added to a previously trained network until no performance improvement is obtained. For smoothness, DeepLab [14] uses a fully-connected CRF to post-process CNN predictions, while the CRF and CNN are iteratively trained, one at a time. Also, a deep deconvolution network presented in [21] uses object proposals as a pre-processing step. For efficiency, we instead use DeconvNet [2], as the number of trainable parameters in DeconvNet is significantly smaller in comparison to peer deep networks.

RTDF is also related to prior work on restricted Boltzmann machine (RBM) [22]. For example, RBMs have been used for extracting both local and global features of object shapes [23], and shape Boltzmann machine (SBM) can generate deformable object shapes [26]. Also, RBM has been used to provide a higher-order potential for a CRF in scene labeling [24, 25].

The most related model to ours is the shape-time random field (STRF) [27]. STRF combines a CRF with a conditional restricted Boltzmann machine (CRBM) [28] for video labeling. They use CRBM to estimate a higher-order potential of the STRF's energy. While this facilitates modeling long-range shape and motion patterns of objects, input to their CRF consists of hand-designed features. Also, they train CRF and CRBM iteratively, as separate modules, in a piece-wise manner. In contrast, we jointly learn all components of our RTDF in a unified manner via loss backpropagation.

3 Recurrent Temporal Deep Field

Our RTDF is an energy-based model that consists of three components – DeconvNet, CRF, and RTRBM – providing the unary, pairwise, and high-order potentials for predicting class labels $\mathbf{y}^t = \{\mathbf{y}_p^t : \mathbf{y}_p^t \in \{0,1\}^L\}$ for pixels p in video frame t, where $\mathbf{y}_p^t$ has only one non-zero element. Labels $\mathbf{y}^t$ are predicted given: (a) pixels $\mathbf{I}^t$ of frame t, and (b) previous RTDF predictions $\mathbf{y}^{<t} = \{\mathbf{y}^{t-1}, \mathbf{y}^{t-2}, \dots, \mathbf{y}^{t-\gamma}\}$, as illustrated in Fig. 2.

DeconvNet takes pixels $\mathbf{I}^t$ as input, and outputs the class likelihoods $\mathbf{x}^t = \{\mathbf{x}_p^t : \mathbf{x}_p^t \in [0,1]^L, \sum_{l=1}^L x_{pl}^t = 1\}$, for every pixel p in frame t. A more detailed description of DeconvNet is given in Sec. 3.3. $\mathbf{x}^t$ is then used to define the unary potential of RTDF.

RTRBM takes previous RTDF predictions $\mathbf{y}^{<t}$ as input and estimates values of latent variables $\mathbf{r}^{<t} = \{\mathbf{r}^{t-1}, \dots, \mathbf{r}^{t-\gamma}\}$ from $\mathbf{y}^{<t}$. The time-unfolded visualization in Fig. 2 shows that $\mathbf{r}^{t-1}$ is affected by previous RTDF predictions $\mathbf{y}^{<t}$ through the full connectivity between two consecutive $\mathbf{r}$ layers and the full connectivity between the corresponding $\mathbf{r}$ and $\mathbf{z}$ layers.

The hidden layer $\mathbf{r}^{t-1}$ is aimed at capturing long-range spatiotemporal dependences of predicted class labels in $\mathbf{y}^{<t}$. Thus, $\mathbf{h}^t$ and $\mathbf{r}^{t-1}$ are used to define the high-order potential of RTDF, which is distributed to all pixels in frame t via the full connectivity between layers $\mathbf{h}^t$ and $\mathbf{z}^t$, as well as between layers $\mathbf{h}^t$ and $\mathbf{r}^{t-1}$ in RTRBM. Specifically, the high-order potential is distributed to each pixel via a deterministic mapping between nodes in $\mathbf{z}^t$ and pixels in $\mathbf{y}^t$. While there are many options for this mapping, in our implementation, we partition frame t into a regular grid of patches. As further explained in Sec. 3.1, each node of $\mathbf{z}^t$ is assigned to a corresponding patch of pixels in $\mathbf{y}^t$.

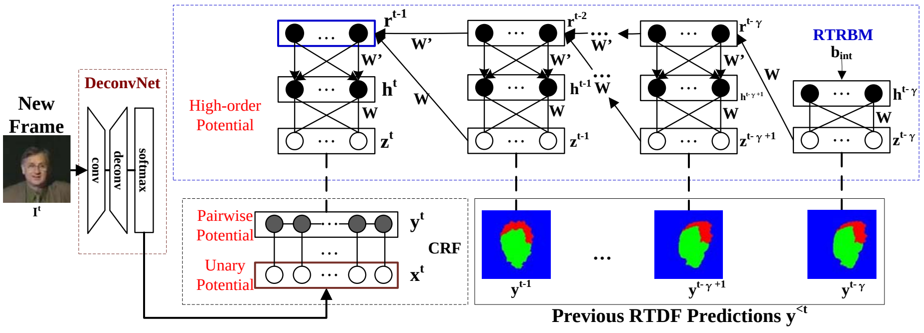


Fig. 2. Our RTDF is an energy-based model that predicts pixel labels $\mathbf{y}^t$ for frame t, given the unary potential $\mathbf{x}^t$ of DeconvNet, the pairwise potential between neighboring pixel labels in frame t, and the high-order potential defined in terms of $\mathbf{z}^t$, $\mathbf{h}^t$ and $\mathbf{r}^{t-1}$ of RTRBM. The figure shows the time-unfolded visualization of computational processes in RTRBM. RTRBM takes as input previous RTDF predictions $\{\mathbf{y}^{t-1}, \ldots, \mathbf{y}^{t-\gamma}\}$ and encodes the long-range and high-order dependencies through latent variables $\mathbf{r}^{t-1}$. The high-order potential is further distributed to all pixels in frame t via a deterministic mapping (vertical dashed lines) between $\mathbf{y}^t$ and $\mathbf{z}^t$.

The energy of RTDF is defined as

$$E_{\mathrm{RTDF}}(\mathbf{y}^t, \mathbf{h}^t | \mathbf{y}^{<t}, \mathbf{I}^t) = -\sum_p \psi_1(\mathbf{x}_p^t, \mathbf{y}_p^t) - \sum_{p,p'} \psi_2(\mathbf{y}_p^t, \mathbf{y}_{p'}^t) + E_{\mathrm{RT}}(\mathbf{y}^t, \mathbf{h}^t | \mathbf{y}^{<t}). \quad (1)$$

In (1), the first two terms denote the unary and pairwise potentials, and the third term represents the high-order potential. As mentioned above, the mapping between $\mathbf{y}^t$ and $\mathbf{z}^t$ is deterministic. Therefore, instead of using $\mathbf{z}^t$ in (1), we can specify E_{RT} directly in terms of $\mathbf{y}^t$. This allows us to conduct *joint* inference of $\mathbf{y}^t$ and $\mathbf{h}^t$, as further explained in Sec. 4.

The unary and pairwise potentials are defined as for standard CRFs:

$$\psi_1(\mathbf{x}_p^t, \mathbf{y}_p^t) = W^1_{\mathbf{y}_p^t} \cdot \mathbf{x}_p^t, \qquad \psi_2(\mathbf{y}_p^t, \mathbf{y}_{p'}^t) = W^2_{\mathbf{y}_p^t, \mathbf{y}_{p'}^t} \cdot \exp(-|\mathbf{x}_p^t - \mathbf{x}_{p'}^t|), \quad (2)$$

where $W^1_{\mathbf{y}} \in \mathbb{R}^L$ is an L-dimensional vector of unary weights for a given class label at pixel p, and $W^2_{\mathbf{y},\mathbf{y}'} \in \mathbb{R}^L$ is an L-dimensional vector of pairwise weights for a given pair of class labels at neighboring pixels p and p'.

Before specifying E_{RT}, for clarity, we first review the restricted Boltzmann machine (RBM) and then explain its extension to RTRBM.

3.1 A Brief Review of Restricted Boltzmann Machine

RTRBM can be viewed as a temporal concatenation of RBMs [3]. RBM [22] is an undirected graphical model with one visible layer and one hidden layer. In our approach, the visible layer consists of L-dimensional binary vectors $\mathbf{z} = \{\mathbf{z}_i : \mathbf{z}_i \in$

$\{0,1\}^L\}$ and each $\mathbf{z}_i$ has only one non-zero element representing the class label of the corresponding patch i in a given video frame. The hidden layer consists of binary variables $\mathbf{h} = \{h_j : h_j \in \{0,1\}\}$. RBM defines a joint distribution of the visible layer $\mathbf{z}$ and the hidden layer $\mathbf{h}$, and the energy function between the two layers for a video frame is defined as:

$$E_{\mathrm{RBM}}(\mathbf{z},\mathbf{h}) = -\sum_j \sum_i \sum_{l=1}^{L} W_{ijl} h_j z_{il} - \sum_i \sum_{l=1}^{L} z_{il} c_{il} - \sum_j b_j h_j \tag{3}$$

where W is the RBM's weight matrix between $\mathbf{z}$ and $\mathbf{h}$, and $\mathbf{b}$ and $\mathbf{c}$ are the bias vectors for $\mathbf{h}$ and $\mathbf{z}$, respectively. RBM has been successfully used for modeling spatial context of an image or video frame [24, 25, 27].

Importantly, to reduce the huge number of parameters in RBM (and thus facilitate learning), we follow the pooling approach presented in [27]. Specifically, instead of working directly with pixels in a video frame, our formulation of RBM uses patches i of pixels (8×8 pixels) as corresponding to the visible variables $\mathbf{z}_i$. The patches are obtained by partitioning the frame into a regular grid.

Recall that in our overall RTDF architecture RBM is grounded onto latent pixel labels $\mathbf{y}_p$ through the deterministic mapping of $\mathbf{z}_i$'s to pixels p that fall within patches i (see Fig. 2). When predicted labels $\mathbf{y}^{<t}$ are available for video frames before time t, we use the following mapping $\mathbf{z}_i = 1/|i| \sum_{p \in i} \mathbf{y}_p$, where $|i|$ denotes the number of pixels in patch i. Note that this will give real-valued $\mathbf{z}_i$'s, which we then binarize. Conversely, for frame t, when we want to distribute the high-order potential, we deterministically assign potential of $\mathbf{z}_i$ to every pixel within the patch.

3.2 A Brief Review of RTRBM

RTRBM represents a recurrent temporal extension of an RBM [3], with one visible layer $\mathbf{z}$, and two hidden layers $\mathbf{h}$ and $\mathbf{r}$. As in RBM, $\mathbf{h}$ are binary variables, and $\mathbf{r} = \{r_j : r_j \in [0,1]\}$ represents a set of real-valued hidden variables. In the time-unfolded visualization shown in Fig. 2, RTRBM can be seen as a temporal concatenation of the respective sets of RBM's variables, indexed by time t, $\{\mathbf{z}^t, \mathbf{h}^t, \mathbf{r}^t\}$. This means that each RBM at time t in RTRBM has a dynamic bias input that is affected by the RBMs of previous time instances. This dynamic bias input is formalized as a recurrent neural network [29], where hidden variables $\mathbf{r}^t$ at time t are obtained as

$$\mathbf{r}^t = \sigma(W\mathbf{z}^t + \mathbf{b} + W'\mathbf{r}^{t-1}), \tag{4}$$

where $\{\mathbf{b}, W, W'\}$ are parameters. Note that $\mathbf{b} + W'\mathbf{r}^{t-1}$ is replaced by $\mathbf{b}_{int}$ for time $t = 1$, $\sigma(\cdot)$ is the element-wise sigmoid function, and W' is the shared weight matrix between $\mathbf{r}^{t-1}$ and $\mathbf{h}^t$ and between $\mathbf{r}^{t-1}$ and $\mathbf{r}^t$. Consequently, the recurrent neural network in RTRBM is designed such that the conditional expectation of $\mathbf{h}^t$, given $\mathbf{z}^t$, is equal to $\mathbf{r}^t$. RTRBM defines an energy of $\mathbf{z}^t$ and

$\mathbf{h}^t$ conditioned on the hidden recurrent input $\mathbf{r}^{t-1}$ as

$$E_{\mathrm{RT}}(\mathbf{z}^t, \mathbf{h}^t|\mathbf{r}^{t-1}) = E_{\mathrm{RBM}}(\mathbf{z}^t, \mathbf{h}^t) - \sum_j \sum_k W'_{jk} h_j^t r_k^{t-1}. \tag{5}$$

From (3), (4) and (5), RTRBM parameters are $\theta_{\mathrm{RT}} = \{\mathbf{b}_{\mathrm{int}}, \mathbf{b}, \mathbf{c}, W, W'\}$. The associated free energy of $\mathbf{z}^t$ is defined as

$$F_{\mathrm{RT}}(\mathbf{z}^t|\mathbf{r}^{t-1}) = -\sum_j \log(1 + \exp(b_j + \sum_{i,l} W_{ijl} z_{il} + \sum_k W'_{jk} r_k^{t-1})) - \sum_{i,l} z_{il} c_{il}. \tag{6}$$

RTRBM can be viewed as capturing long-range and high-order dependencies in both space and time, because it is characterized by the full connectivity between consecutive $\mathbf{r}$ layers, and between the corresponding $\mathbf{r}$, $\mathbf{z}$, and $\mathbf{h}$ layers.

Due to the deterministic mapping between $\mathbf{z}^t$ and $\mathbf{y}^t$ for frame t, we can specify E_{RT} given by (5) in terms of $\mathbf{y}^t$, i.e., as $E_{\mathrm{RT}}(\mathbf{y}^t, \mathbf{h}^t|\mathbf{r}^{t-1})$. We will use this to derive a mean-field inference of $\mathbf{y}^t$, as explained in Sec. 4.

3.3 DeconvNet

As shown in Fig. 2, DeconvNet [2] is used for computing the unary potential of RTDF. We strictly follow the implementation presented in [2]. DeconvNet consists of two networks: one based on VGG16 net to encode the input video frame, and a multilayer deconvolution network to generate feature maps for predicting pixel labels. The convolution network records the pooling indices computed in the pooling layers. Given the output of the convolution network and the pooling indices, the deconvolution network performs a series of unpooling and deconvolution operations for producing the final feature maps. These feature maps are passed through the softmax layer for predicting the likelihoods of class labels of every pixel, $\mathbf{x}_p \in [0,1]^L$. Before joint training, we pre-train parameters of DeconvNet, θ_{DN}, using the cross entropy loss, as in [2].

4 Inference of RTDF

Pixel labels of the first γ frames of a video are *predicted* using a variant of our model – namely, the jointly trained CRF + DeconvNet, without RTRBM. Then, inference of the full RTDF (i.e., jointly trained CRF + DeconvNet + RTRBM) proceeds to subsequent frames until all the frames have been labeled.

Given a sequence of semantic labelings in the past, $\mathbf{y}^{<t}$, and a new video frame, $\mathbf{I}^t$, the goal of RTDF inference is to predict $\mathbf{y}^t$ as:

$$\hat{\mathbf{y}}^t = \arg\max_{\mathbf{y}^t} \sum_{\mathbf{h}^t} \exp(-E_{\mathrm{RTDF}}(\mathbf{y}^t, \mathbf{h}^t|\mathbf{y}^{<t}, \mathbf{I}^t)). \tag{7}$$

Since the exact inference of RTDF is intractable, we formulate an approximate mean-field inference for jointly predicting both $\hat{\mathbf{y}}^t$ and $\hat{\mathbf{h}}^t$. Its goal is to minimize the KL-divergence between the true posterior distribution, $P(\mathbf{y}^t, \mathbf{h}^t|\mathbf{y}^{<t}, \mathbf{I}^t)$

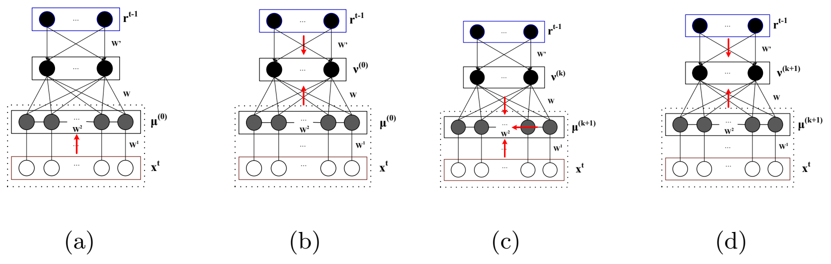


Fig. 3. Key steps of the mean-field inference overlaid over RTDF which is depicted as in Fig. 1b. (a) Initialization of $\boldsymbol{\mu}^{(0)}$. (b) Initialization of $\boldsymbol{\nu}^{(0)}$. (c) Updating of $\boldsymbol{\mu}^{(k+1)}$. (d) Updating of $\boldsymbol{\nu}^{(k+1)}$. The red arrows show the information flow.

$= \frac{1}{Z(\theta)} \exp(-E_{\text{RTDF}}(\mathbf{y}^t, \mathbf{h}^t | \mathbf{y}^{<t}, \mathbf{I}^t))$, and the mean-field distribution $Q(\mathbf{y}^t, \mathbf{h}^t) = \prod_p Q(\mathbf{y}_p^t) \prod_j Q(\mathbf{h}_j^t)$ factorized over pixels p for $\mathbf{y}^t$ and hidden nodes j for $\mathbf{h}^t$.

To derive our mean-field inference, we introduce the following two types of variational parameters: (i) $\boldsymbol{\mu} = \{\mu_{pl} : \mu_{pl} = Q(\mathbf{y}_{pl}^t = 1)\}$, where $\sum_{l=1}^{L} \mu_{pl} = 1$ for every pixel p; (ii) $\boldsymbol{\nu} = \{\nu_j : \nu_j = Q(\mathbf{h}_j^t = 1)\}$. They allow us to express the mean-field distribution as $Q(\mathbf{y}^t, \mathbf{h}^t) = Q(\boldsymbol{\mu}, \boldsymbol{\nu}) = \prod_p \mu_p \prod_j \nu_j$. It is straightforward to show that minimizing the KL-divergence between P and Q amounts to the following objective

$$\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\nu}} = \arg\max_{\boldsymbol{\mu},\boldsymbol{\nu}} \{ \sum_{\mathbf{y}^t, \mathbf{h}^t} Q(\boldsymbol{\mu}, \boldsymbol{\nu}) \ln P(\mathbf{y}^t, \mathbf{h}^t | \mathbf{y}^{<t}, \mathbf{I}^t) + H(Q(\boldsymbol{\mu}, \boldsymbol{\nu})) \}, \tag{8}$$

where $H(Q)$ is the entropy of Q.

Our mean-field inference begins with initialization: $\mu_{pl}^{(0)} = \frac{\exp(W^1_{\mu_{pl}} \cdot \mathbf{x}_p)}{\sum_{l'} \exp(W^1_{\mu_{pl'}} \cdot \mathbf{x}_p)}$, $\nu_j^{(0)} = \sigma(\sum_l \sum_i \sum_{p \in i} \frac{1}{|i|} \mu_{pl}^{(0)} W_{ijl} + b_j + \sum_{j'} W'_{jj'} r_{j'}^{t-1})$ and then proceeds by updating $\mu_{pl}^{(k)}$ and $\nu_j^{(k)}$ using the following equations until convergence:

$$\mu_{pl}^{(k+1)} = \frac{\exp(W^1_{\mu_{pl}^{(k)}} \cdot \mathbf{x}_p + \sum_j W_{ijl} \nu_j^{(k)} + c_{il} + \beta_{p' \to p}^{(k)})}{\sum_{l'} \exp(W^1_{\mu_{pl'}^{(k)}} \cdot \mathbf{x}_p + \sum_j W_{ijl'} \nu_j^{(k)} + c_{il'} + \beta_{p' \to p}^{(k)})}, \tag{9}$$

$$\nu_j^{(k+1)} = \sigma(\sum_l \sum_i \sum_{p \in i} \frac{1}{|i|} \mu_{pl}^{(k+1)} W_{ijl} + b_j + \sum_{j'} W'_{jj'} r_{j'}^{t-1}), \tag{10}$$

where $\beta_{p' \to p}^{(k)} = \sum_{p'} \sum_{l'} W^2_{\mu_{pl}^{(k)}, \mu_{p'l'}^{(k)}} \cdot \exp(-|\mathbf{x}_p - \mathbf{x}_{p'}|)$ denotes a pairwise term that accounts for all neighbors p' of p, W^1 and W^2 denote parameters of the unary and pairwise potentials defined in (2), and W_{ijl} and $W'_{jj'}$ are parameters of RTRBM. Also, the second and the third terms in (9) and the first term in (10) use the deterministic mapping between patches i and pixels $p \in i$ (see

Algorithm 1: Joint Training of RTDF

```
input  : Training set: ${\mathbf{I}^t, \mathbf{y}^t, t = 1, 2, \cdots}$, where $\mathbf{y}^t$ is ground truth
output: Parameters of RTDF
repeat
    1. For every training video, conduct the mean-field inference, presented in
    Sec.4, and calculate the free energy associated with $\mathbf{y}^t$ using (12);
    2. Compute the derivative of $\triangle(\theta)$ given by (11) with respect to:
       2.1. Unary term $\mathbf{x}_p$, using (11) and (2),
       2.2. Pairwise term $\exp(-|\mathbf{x}_p^t - \mathbf{x}_{p'}^t|)$, using (11) and (2);
    3. Update CRF parameters $W^1$, $W^2$, using the result of Step 2;
    4. Backpropagate the result of Step 2.1. to DeconvNet using the chain rule
    in order to update $\theta_{DN}$;
    5. Compute $\frac{\partial \triangle}{\partial \theta_{\mathrm{RT}}}$ using (11), (12), (6) and (4) for updating $\theta_{\mathrm{RT}}$;
until stopping criteria;
```

Sec. 3.1). Fig. 3 shows the information flow in our mean-field inference, overlaid over RTDF which is depicted in a similar manner as in Fig. 1b.

After convergence at step K, the variational parameter $\boldsymbol{\mu}^{(k)}$, $k \in \{0, 1, \cdots, K\}$ associated with minimum free energy as defined in (12) is used to predict the label of pixels in frame t. The label at every pixel p is predicted as l for which $\mu_{pl}^{(k)}$, $l \in \{1, 2, \cdots, L\}$ is maximum. This amounts to setting $\hat{y}_{pl}^t = 1$, while all other elements of vector $\hat{\mathbf{y}}_p^t$ are set to zero. Also, the value of $\hat{h}_j^t$ is estimated by binarizing the corresponding maximum $\nu_j^{(k)}$.

5 Learning

Parameters of all components of RTDF, $\theta = \{W^1, W^2, \theta_{\mathrm{DN}}, \theta_{\mathrm{RT}}\}$, are trained jointly. For a suitable initialization of RTDF, we first pretrain each component, and then carry out joint training, as summarized in Alg. 1.

Pretraining. (1) **RTRBM.** The goal of learning RTRBM is to find parameters θ_{RTRBM} that maximize the joint log-likelihood, $\log p(\mathbf{z}^{<t}, \mathbf{z}^t)$. To this end, we closely follow the learning procedure presented in [3], which uses the backpropagation-through-time (BPTT) algorithm [29] for back-propagating the error of patch labeling. As in [3], we use contrastive divergence (CD) [30] to approximate the gradient in training RTRBM. (2) **DeconvNet.** As initial parameters, DeconvNet uses parameters of VGG16 network (without the fully-connected layers) for the deep convolution network, and follows the approach of [2] for the deconvolution network. Then, the two components of DeconvNet are jointly trained using the cross entropy loss defined on pixel label predictions. (3) **CRF.** The CRF is pretrained on the output features from DeconvNet using loopy belief propagation with the LBFGS optimization method.

Joint Training of RTDF. The goal of joint training is to maximize the conditional log-likelihood $\sum_t \log p(\mathbf{y}^t|\mathbf{y}^{<t}, \mathbf{I}^t)$. We use CD-PercLoss algorithm

[31] and error back-propagation (EBP) to jointly train parameters of RTDF in an end-to-end fashion. The training objective is to minimize the following generalized perceptron loss [32] with regularization:

$$\triangle(\theta) = \sum_t (F(\mathbf{y}^t|\mathbf{y}^{<t}, \mathbf{I}^t) - \min_{\hat{\mathbf{y}}^t} F(\hat{\mathbf{y}}^t|\mathbf{y}^{<t}, \mathbf{I}^t)) + \lambda\theta^T\theta \tag{11}$$

where $\lambda > 0$ is a weighting parameter, and $F(\mathbf{y}^t|\mathbf{y}^{<t}, \mathbf{I}^t)$ denotes the free energy of ground truth label $\mathbf{y}^t$ of frame t, and $\hat{\mathbf{y}}^t$ is the predicted label associated with minimum free energy. The free energy of RTDF is defined as

$$F(\mathbf{y}^t|\mathbf{y}^{<t}, \mathbf{I}^t) = -\sum_p \psi_1(\mathbf{x}_p^t, \mathbf{y}_p^t) - \sum_{p,p'} \psi_2(\mathbf{y}_p^t, \mathbf{y}_{p'}^t) + F_{\mathrm{RT}}(\mathbf{y}^t|\mathbf{r}^{t-1}) \tag{12}$$

where the first two terms denote the unary and pairwise potentials, and the third term is defined in (6). In the prediction pass of training, the pixel label is obtained by the mean-field inference, as explained in Sec.4. In the updating phase of training, the errors are back-propagated through CRF, DeconvNet and RTRBM in a standard way, resulting in a joint update of θ.

6 Results

Datasets and Metrics: For evaluation, we use the Youtube Face Database (YFDB) [1] and Cambridge-driving Labeled Video Database (CamVid) [33]. Both datasets are recorded in uncontrolled environment, and present challenges in terms of occlusions, and variations of motions, shapes, and lighting. CamVid consists of four long videos showing driving scenes with various object classes, whose frequency of appearance is unbalanced. Unlike other available datasets [34–36], YFDB and CamVid provide sufficient training samples for learning RTRBM. Each YFDB video contains 49 to 889 roughly aligned face images with resolution 256 × 256. We use the experimental setup of [27] consisting of randomly selected 50 videos from YFDB, with ground-truth labels of hair, skin, and background provided for 11 consecutive frames per each video (i.e., 550 labeled frames), which are then split into 30, 10, and 10 videos for training, validation, and testing, respectively. Each CamVid video contains 3600 to 11000 frames at resolution 360 × 480. CamVid provides ground-truth pixel labels of 11 object classes for 700 frames, which are split into 367 training and 233 test frames. For fair comparison on CamVid with [2], which uses significantly more training data, we additionally labeled 9 consecutive frames preceding every annotated frame in the training set of CamVid, resulting in 3670 training frames.

For fair comparison, we evaluate our superpixel accuracy on YFDB and pixel accuracy on Camvid. For YFDB, we extract superpixels as in [27] producing 300-400 superpixels per frame. The label of a superpixel is obtained by pixel majority voting. Both overall accuracy and class-specific accuracy are computed as the number of superpixels/pixels classified correctly divided by the total number of superpixels/pixels. Evaluation is done for each RTDF prediction on a test frame

Table 1. Superpixel accuracy on Youtube Face Database [1]. Error reduction in overall superpixel accuracy is cacluated w.r.t the CRF. The mean and the standard derivation are given from a 5-fold cross-validation.

Model	Error Redu	Overall Accu	Hair	Skin	Background	Category Avg
CRF [4]	0.0	0.90 ± 0.005	0.63 ± 0.047	0.89 ± 0.025	0.96 ± 0.005	0.83 ± 0.009
GLOC [25]	0.03 ± 0.025	0.91 ± 0.006	0.61 ± 0.038	0.90 ± 0.023	0.96 ± 0.003	0.82 ± 0.008
STRF [27]	0.12 ± 0.025	0.91 ± 0.006	0.72 ± 0.039	0.89 ± 0.025	0.96 ± 0.004	0.86 ± 0.010
RTDF†	0.11 ± 0.027	0.91 ± 0.008	0.70 ± 0.043	0.89 ± 0.024	0.96 ± 0.004	0.85 ± 0.011
RTDF*	0.17 ± 0.028	0.92 ± 0.008	0.76 ± 0.049	0.88 ± 0.025	0.96 ± 0.003	0.87 ± 0.012
RTDF	**0.34 ± 0.031**	**0.93 ± 0.010**	**0.80 ± 0.037**	**0.90 ± 0.026**	**0.97 ± 0.005**	**0.89 ± 0.014**

after processing 3 and 4 frames preceding that test frame for YFDB and Camvid, respectively.

Implementation Details: We partition video frames using a 32×32 regular grid for YFDB, and a 60×45 regular grid for CamVid. For YFDB, we specify RTRBM with 1000 hidden nodes. For CamVid, there are 1200 hidden nodes in RTRBM. Hyper-parameters of the DeconvNet are specified as in [2]. The DeconvNet consists of: (a) Convolution network with 13 convolution layers based on VGG16 network, each followed by a batch normalization operation [42] and a RELU layer; (b) Deconvolution network with 13 deconvolution layers, each followed by the batch normalization; and (c) Soft-max layer producing a $1 \times L$ class distribution for every pixel in the image. We test $\lambda \in [0, 1]$ on the validation set, and report our test results for λ with the best performance on the validation dataset.

Runtimes: We implement RTDF on NVIDIA Tesla K80 GPU accelerator. It takes about 23 hours to train RTDF on CamVid. The average runtime for predicting pixel labels in an image with resolution 360×480 is 105.3ms.

Baselines: We compare RTDF with its variants and related work: 1) RTDF†: RTDF without end-to-end joint training (i.e., piece wise training); 2) RTDF*: jointly trained RTDF without joint inference, i.e., using stage-wise inference where the output of RTRBM is treated as fixed input into the CRF. 3) CRF [4]: spatial CRF inputs with hand-engineered features; 4) GLOC [25]: a jointly trained model that combines spatial CRF and RBM; and 5) STRF [27]: a piece-wise trained model that combines spatial CRF, CRBM and temporal potentials between two consecutive frames.

6.1 Quantitative Results

YFDB: Tab. 1 presents the results of the state of the art, RTDF and its variant baselines on YFDB. As can be seen, RTDF gives the best performance, since RTDF accounts for long-range spatiotemporal dependencies and performs joint training and joint inference. It outperforms STRF [27] which uses local hand-engineered features and piece-wise training. These results suggest that accounting for object interactions across a wide range of spatiotemporal scales is critical for video labeling. We also observe that RTDF† achieves comparable results with STRF [27], while RTDF* outperforms both. This suggests that our

Table 2. Pixel accuracy on Cambridge-driving Labeled Video Database [33].

Method	Building	Tree	Sky	Car	Sign	Road	Pedestrain	Fence	Col. Pole	Sidewalk	Bicycle	Class Avg.	Global Avg.
Dense Depth Maps [37]	85.3	57.3	95.4	69.2	46.5	**98.5**	23.8	44.3	22.0	38.1	28.7	55.4	82.1
Super Parsing [38]	87.0	67.1	96.9	62.7	30.1	95.9	14.7	17.9	1.7	70.0	19.4	51.2	83.3
High-order CRF [39]	84.5	72.6	**97.5**	72.7	34.1	95.3	34.2	45.7	8.1	77.6	28.5	59.2	83.8
CRF + Detectors [40]	81.5	76.6	96.2	78.7	40.2	93.9	43.0	47.6	14.3	81.5	33.9	62.5	83.8
Neural Decision Forests [41]	N/A											56.1	82.1
Deeplab [14]	82.7	**91.7**	89.5	76.7	33.7	90.8	41.6	35.9	17.9	82.3	45.9	62.6	84.6
CRFasRNN [20]	84.6	91.3	92.4	79.6	43.9	91.6	37.1	36.3	27.4	82.9	33.7	63.7	86.1
SegNet [2]	73.9	90.6	90.1	86.4	**69.8**	94.5	**86.8**	**67.9**	**74.0**	94.7	52.9	80.1	86.7
RTDF†	81.8	87.9	91.5	79.2	59.8	90.4	77.1	61.5	66.6	91.2	54.6	76.5	86.5
RTDF*	83.6	89.8	92.9	78.5	61.3	92.2	79.6	61.9	67.7	92.8	56.9	77.9	88.1
RTDF	**87.1**	85.2	93.7	**88.3**	64.3	94.6	84.2	64.9	68.8	**95.3**	**58.9**	**80.5**	**89.9**

end-to-end joint training of all components of RTDF is more critical for accurate video labeling than their joint inference. Also, as RTDF* gives an inferior performance to RTDF, performing joint instead of stage-wise inference gives an additional gain in performance. Finally, we observe that RTDF performance can be slightly increased by using a larger γ. For fair comparison, we use the same γ as in [27].

CamVid: Tab. 2 presents the results of the state of the art, RTDF and its variants on CamVid. In comparison to the state of the art and the baselines, RTDF achieves superior performance in terms of both average and weighted accuracy, where weighted accuracy accounts for the class frequency. Unlike RTDF, SegNet [2] treats the label of each pixel independently by using a soft-max classifier, and thus may poorly perform around low-contrast object boundaries. On the other hand, SegNet has an inherent bias to label larger pixel areas with a unique class label [2] (see Fig. 4), which may explain its better performance than RTDF on the following classes: sign-symbol, column-pole, pedestrian and fence. From Tab. 2, RTDF† achieves comparable performance to that of SegNet, while RTDF* outperforms RTDF†. This is in agreement with our previous observation on YFDB that joint training of all components of RTDF is more critical than their joint inference for accurate video labeling.

6.2 Qualitative Evaluation

Fig. 4 illustrates our pixel-level results on frame samples of CamVid. From the figure, we can see that our model is able to produce spatial smoothness pixel labeling. Fig. 6 shows superpixel labeling on sample video clips from YFDB. As can be seen, on both sequences, STRF [27] gives inferior video labeling than RTDF in terms of temporal coherency and spatial consistency of pixel labels. Our spatial smoothness and temporal coherency can also be seen in Fig. 5 which shows additional RTDF results on a longer sequence of frames from a sample CamVid video.

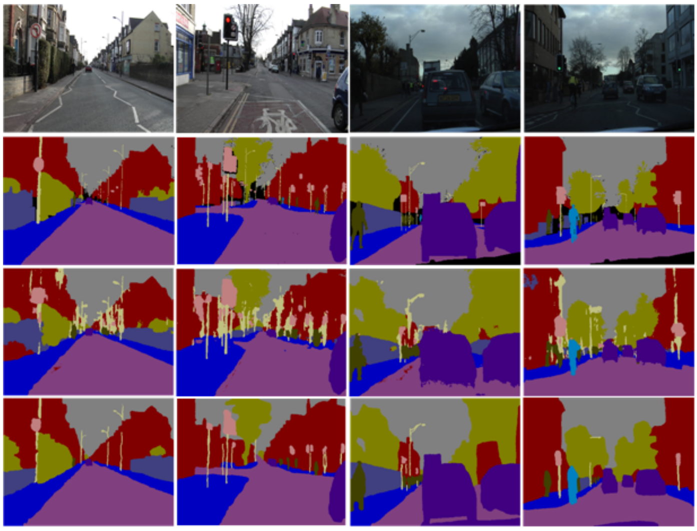


Fig. 4. Frame samples from CamVid. The rows correspond to original images, ground truth, SegNet [2], and RTDF.

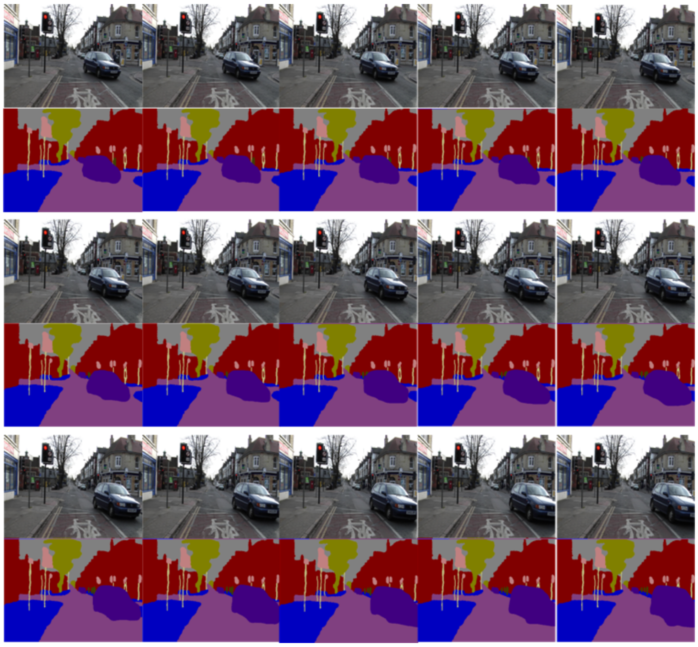


Fig. 5. Sequence of frames from a sample CamVid video. The rows correspond to input frames and RTDF outputs.

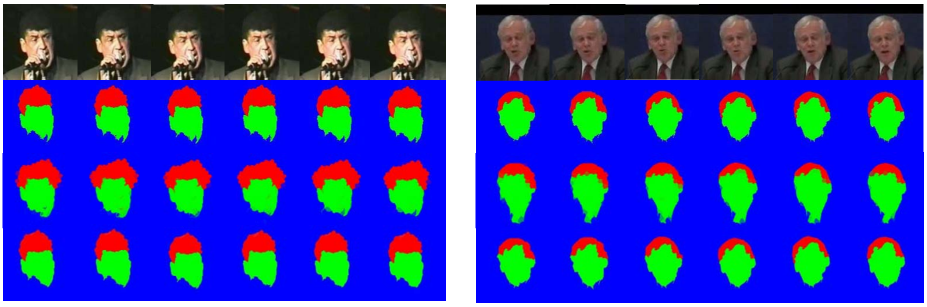


Fig. 6. Frame sequences from two CamVid video clips. The rows correspond to original video frames, ground truth, STRF[27], and RTDF.

Empirically, we find that RTDF poorly handles abrupt scale changes (e.g., dramatic camera zoom-in/zoom-out). Also, in some cases shown in Fig. 4 and Fig. 5, RTDF misses tiny, elongated objects like column-poles, due to our deterministic mapping between patches of a regular grid and pixels.

7 Conclusion

We have presented a new deep architecture, called Recurrent-Temporal Deep Field (RTDF), for semantic video labeling. RTDF captures long-range and high-order spatiotemporal dependencies of pixel labels in a video by combining conditional random field (CRF), deconvolution neural network (DeconvNet), and recurrent temporal restricted Boltzmann machine (RTRBM) into a unified framework. Specifically, we have derived a mean-field inference algorithm for jointly predicting latent variables in both CRF and RTRBM, and specified an end-to-end joint training of all components of RTDF via backpropagation of the prediction loss. Our empirical evaluation on the benchmark Youtube Face Database (YFDB) [1] and Cambridge-driving Labeled Video Database (CamVid) [33] demonstrates the advantages of performing joint inference and joint training of RTDF, resulting in its superior performance over the state of the art. The results suggest that our end-to-end joint training of all components of RTDF is more critical for accurate video labeling than their joint inference. Also, RTDF performance on a frame can be improved by previously labeling longer sequences of frames preceding that frame. Finally, we have empirically found that RTDF poorly handles abrupt scale changes and labeling of thin, elongated objects.

Acknowledgment

This work was supported in part by grant NSF RI 1302700. The authors would like to thank Sheng Chen for useful discussion and acknowledge Dimitris Trigkakis for helping with the datasets.

References

1. Wolf, L., Hassner, T., Maoz, I.: Face recognition in unconstrained videos with matched background similarity. In: CVPR. (2011)
2. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. arXiv preprint arXiv:1511.00561 (2015)
3. Sutskever, I., Hinton, G.E., Taylor, G.W.: The recurrent temporal restricted boltzmann machine. In: NIPS. (2009)
4. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: ICML. (2001)
5. Galmar, E., Athanasiadis, T., Huet, B., Avrithis, Y.: Spatiotemporal semantic video segmentation. In: MSPW. (2008)
6. Grundmann, M., Kwatra, V., Han, M., Essa, I.: Efficient hierarchical graph-based video segmentation. In: CVPR. (2010)
7. Jain, A., Chatterjee, S., Vidal, R.: Coarse-to-fine semantic video segmentation using supervoxel trees. In: ICCV. (2013)
8. Yi, S., Pavlovic, V.: Multi-cue structure preserving mrf for unconstrained video segmentation. arXiv preprint arXiv:1506.09124 (2015)
9. Zhao, H., Fu, Y.: Semantic single video segmentation with robust graph representation. In: IJCAI. (2015)
10. Liu, B., He, X., Gould, S.: Multi-class semantic video segmentation with exemplar-based object reasoning. In: WACV. (2015)
11. Taylor, B., Ayvaci, A., Ravichandran, A., Soatto, S.: Semantic video segmentation from occlusion relations within a convex optimization framework. In: EMMCVPR. (2013)
12. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. PAMI **35**(8) (2013) 1915–1929
13. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR. (2015)
14. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected crfs. In: ICLR. (2014)
15. Ciresan, D., Giusti, A., Gambardella, L.M., Schmidhuber, J.: Deep neural networks segment neuronal membranes in electron microscopy images. In: NIPS. (2012)
16. Pinheiro, P.H., Collobert, R.: Recurrent convolutional neural networks for scene parsing. In: ICML. (2014)
17. Hariharan, B., Arbeláez, P., Girshick, R., Malik, J.: Simultaneous detection and segmentation. In: ECCV. (2014)
18. Gupta, S., Girshick, R., Arbeláez, P., Malik, J.: Learning rich features from rgb-d images for object detection and segmentation. In: ECCV. (2014)
19. Ganin, Y., Lempitsky, V.: Nˆ4-fields: Neural network nearest neighbor fields for image transforms. In: ACCV. (2014)
20. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.: Conditional random fields as recurrent neural networks. In: ICCV. (2015)
21. Noh, H., Hong, S., Han, B.: Learning deconvolution network for semantic segmentation. In: ICCV. (2015)
22. Smolensky, P.: Information processing in dynamical systems: Foundations of harmony theory. MIT Press Cambridge (1986)

23. He, X., Zemel, R.S., Carreira-Perpiñán, M.Á.: Multiscale conditional random fields for image labeling. In: CVPR. (2004)
24. Li, Y., Tarlow, D., Zemel, R.: Exploring compositional high order pattern potentials for structured output learning. In: CVPR. (2013)
25. Kae, A., Sohn, K., Lee, H., Learned-Miller, E.: Augmenting crfs with boltzmann machine shape priors for image labeling. In: CVPR. (2013)
26. Eslami, S.A., Heess, N., Williams, C.K., Winn, J.: The shape boltzmann machine: a strong model of object shape. IJCV **107**(2) (2014) 155–176
27. Kae, A., Marlin, B., Learned-Miller, E.: The shape-time random field for semantic video labeling. In: CVPR. (2014)
28. Taylor, G.W., Hinton, G.E., Roweis, S.T.: Modeling human motion using binary latent variables. In: NIPS. (2006)
29. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. Technical report, DTIC Document (1985)
30. Hinton, G.E.: Training products of experts by minimizing contrastive divergence. Neural Computation (2002)
31. Mnih, V., Larochelle, H., Hinton, G.E.: Conditional restricted boltzmann machines for structured output prediction. In: UAI. (2011)
32. LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., Huang, F.: A tutorial on energy-based learning. Predicting structured data **1** (2006) 0
33. Brostow, G.J., Fauqueur, J., Cipolla, R.: Semantic object classes in video: A high-definition ground truth database. PRL (2008)
34. Li, F., Kim, T., Humayun, A., Tsai, D., Rehg, J.M.: Video segmentation by tracking many figure-ground segments. In: ICCV. (2013)
35. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: CVPR. (2012)
36. Brox, T., Malik, J.: Object segmentation by long term analysis of point trajectories. In: ECCV. (2010)
37. Zhang, C., Wang, L., Yang, R.: Semantic segmentation of urban scenes using dense depth maps. In: ECCV. (2010)
38. Tighe, J., Lazebnik, S.: Superparsing. IJCV **101**(2) (2013) 329–349
39. Sturgess, P., Alahari, K., Ladicky, L., Torr, P.H.: Combining appearance and structure from motion features for road scene understanding. In: BMVC. (2009)
40. Ladickỳ, L., Sturgess, P., Alahari, K., Russell, C., Torr, P.H.: What, where and how many? combining object detectors and CRFs. In: ECCV. (2010)
41. Rota Bulo, S., Kontschieder, P.: Neural decision forests for semantic image labelling. In: CVPR. (2014)
42. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML. (2015)