Budget-Aware Deep Semantic Video Segmentation

Behrooz Mahasseni, Sinisa Todorovic and Alan Fern
Oregon State University
Corvallis, OR
behrooz.mahasseni@gmail.com {sinisa,alan.fern}@oregonstate.edu

Abstract

In this work, we study a poorly understood trade-off between accuracy and runtime costs for deep semantic video segmentation. While recent work has demonstrated advantages of learning to speed-up deep activity detection, it is not clear if similar advantages will hold for our very different segmentation loss function, which is defined over individual pixels across the frames. In deep video segmentation, the most time consuming step represents the application of a CNN to every frame for assigning class labels to every pixel, typically taking 6-9 times of the video footage. This motivates our new budget-aware framework that learns to optimally select a small subset of frames for pixelwise labeling by a CNN, and then efficiently interpolates the obtained segmentations to yet unprocessed frames. This interpolation may use either a simple optical-flow guided mapping of pixel labels, or another significantly less complex and thus faster CNN. We formalize the frame selection as a Markov Decision Process, and specify a Long Short-Term Memory (LSTM) network to model a policy for selecting the frames. For training the LSTM, we develop a policy-gradient reinforcement-learning approach for approximating the gradient of our non-decomposable and non-differentiable objective. Evaluation on two benchmark video datasets show that our new framework is able to significantly reduce computation time, and maintain competitive video segmentation accuracy under varying budgets.

1. Introduction

We consider the problem of semantic video segmentation, where the goal is to assign a correct class label to every pixel in a video. Recently deep networks have achieved state-of-the-art results for semantic video segmentation [32, 35, 23, 48, 10], where typically a Convolutional Neural Network (CNN) is applied to directly label each pixel of each frame. In this way, they significantly improve on the the processing time of more traditional approaches (e.g., energy minimization), as the latter requires time for graphical-model inference and pre-processing time for feature extraction. However, despite the feed-forward architecture of CNNs, and their parallelizable computation on GPUs, runtimes are still far from real time. For example, it takes 6-9 times the video length to execute the approach of [2] on benchmark datasets. Unfortunately, this rules out the practical use of these approaches for applications with tighter runtime constraints. In some situations hardware solutions can help meet the constraints. However, it is equally important to develop a better understanding of how to achieve maximal accuracy given constraints on the compute time and resources.

In this paper, we address the above problem by introducing a framework for budget-aware deep semantic video segmentation. Our approach is applicable to any available semantic segmentation network, which is treated as a black box. Given a semantic segmentation network and a time budget, our approach attempts to maximize accuracy within the budget. The main idea is based on the observation that videos typically show smooth motions, and hence pixel labels of a frame can be efficiently and accurately interpolated from neighboring segmentations. This motivates intelligently selecting frames to be processed by a deep segmentation network and then using fast interpolation networks to assign pixel values to unselected frames. When the interpolation network is significantly faster than the segmentation network there can be significant computational savings.

The problem of budget aware inference is receiving increasing attention as state-of-the-art accuracies approach the needs of many applications. Recently, this problem has been studied for activity detection in video [47], where the goal is to efficiently identify activities using deep networks while avoiding the need to process all video frames. In that work, a recurrent "attention model" was learned which aimed to select a small subset of the video frames for processing, while maintaining high accuracy. The results showed that high accuracies could be maintained while only processing approximately 2% of the video frames. While this result suggest that intelligent frame selection is a viable

way to speedup deep architectures, the problem of activity detection is quite different from our problem of semantic segmentation. In particular, unlike activity detection, the loss function for semantic segmentation is defined over all pixels of a video and it is necessary to assign predicted values to all pixels. This raises the question of what accuracy-time tradeoffs can be achieved for semantic segmentation.

Three Key Ideas: We extend the state of the art by developing: (1) A deep visual attention model, represented as an LSTM network, aimed at optimally selecting only a subset of video frames whose total time of processing by a segmentation network would not exceed a budget constraint. (2) A fast interpolation model, represented by a one-layer CNN, for efficiently labeling the remaining unprocessed frames based on neighboring frames selected for segmentation by the attention model; and (3) Joint learning of main components of our approach – namely, the attention and interpolation models, such that accuracy of the resulting video segmentation is maximized for a given budget.

Fig. 1 illustrates the two stages of our approach. In Stage 1 (Fig. 1 left), the LSTM policy is run to select T frames for which to apply the segmentation network f. In Stage 2 (Fig. 1 right), the interpolation model $\boldsymbol{g}$ is applied to the remaining video frames. As shown in Sec. 4, the total execution time of our approach is a strictly increasing function of T. Hence, from the constraint that our total runtime should be less than the budget B, we can easily estimate the optimal T, by simply stopping the LSTM policy before we exceed the budget.

Our joint training approach is based on recognizing that our problem is inherently one of sequential decision making. Accordingly we draw on ideas from reinforcement learning, which is an area that focuses on learning for sequential decision making. In particular, we derive a policy-gradient reinforcement learning algorithm for our problem, which is shown to be an effective approach for training our models.

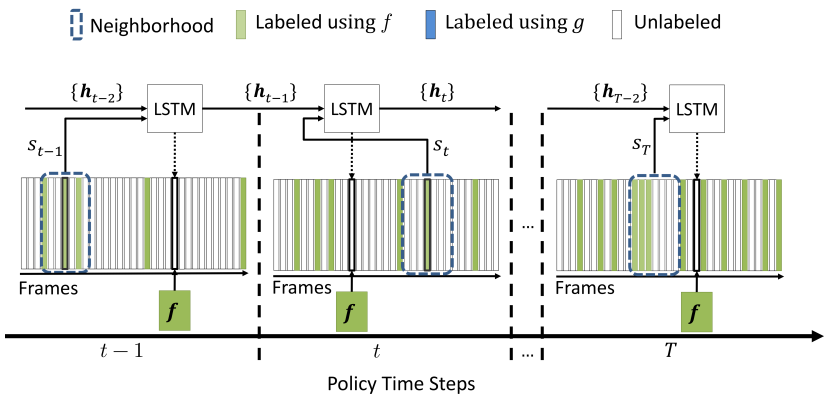
In Sec. 5 we evaluate our approach for varying time budgets and two different semantic segmentation netoworks. Our results show that for budgets $\frac{1}{4}B_{\text{max}} \leq B \leq \frac{1}{2}B_{\text{max}}$, we achieve semantic segmentation accuracy that is comparable to the results obtained by directly segmenting each frame. This shows that our approach is able to learn to speedup the segmentation performance by a factor of four, with little loss in accuracy. Moreover, our accuracy gracefully degrades for $B < \frac{1}{4}B_{\text{max}}$, which is important for applications with very tight budget constraints.

2. Related Work

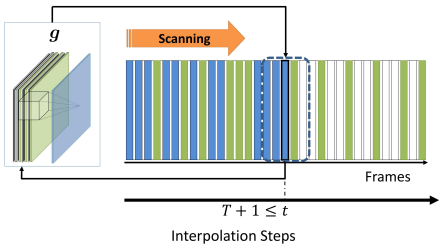
Semantic video segmentation is a long standing problem, and a thorough review of the literature is beyond our scope. It has been traditionally formulated as an energy minimization of a graphical model representing supervoxels or superpixels in the video [7, 6, 5, 33, 8, 29, 39, 45, 31, 20]. Except for a few empirical results of sensitivity to the total number of supervoxels [19, 28] or greedy feature selection in [17], these approaches usually do not explicitly study trade-offs between accuracy and efficiency under varying time constraints. Our main hypothesis is that knowing the budget constraint in training provides additional information which allows the learning algorithm to optimize its decisions toward maximizing overall labeling accuracy. More importantly, it is hard to adapt the approaches proposed in [19, 28, 17] for deep learning base semantic segmentation models.

Recent work on CNN-based semantic segmentation [2, 23, 32, 18, 35, 50, 9, 26, 36, 38] does not require unsupervised segmentation in a pre-processing step, but directly take pixels of the image as input and output a semantic segmentation. These approaches first use a set of (convolution + pooling) layers to generate a deep feature for the entire image. Then, they perform a sequence of deconvolution + upsampling operations for generating an output feature map. As these approaches [2, 23, 32, 18, 35, 50, 9, 26, 36, 38] conduct CNN-based segmentation for every frame independently, their pixel labeling is typically *not* spatiotemporally smooth and coherent. Recently, Peng el al. [25] propose a recurrent temporal field model which considers smoothness of labels is space-time. Also, the runtime of these feed-forward CNN architectures is typically 6-9 times the video length [35].

Efficient inference under budget has recently received much traction in many areas of computer vision [49, 24, 4, 1, 21, 22, 46, 41, 42, 37, 27, 17, 3]. These approaches typically model a utility function of their inference steps, and economically run those steps with the maximum utility. Our key difference is in that we directly learn a budget aware inference policy that achieves high utility within the deep learning framework. These approaches are not easy to generalize to the state-of-the-art CNN based semantic segmentation, since such approaches do not explicitly extract features during a preprocessing step. Unlike these approaches we do not require extracting additional features for estimating the utility, but instead train our deep budget-aware semantic segmentation model in an end-to-end fashion. Vijayanarasimhan et al. [40] use the idea of informative frame selection and label propagation to facilitate human annotation in semantic video segmentation in an active learning framework. Their work is different than ours in: 1) They approach is built on top of the traditional CRF based semantic labeling, 2) Their full model requires solving another CRF model which takes at least a minute itself, and 3) They are trying to improve the human labeling time which is in the order of 25 minutes per frame. Mahasseni et al. [3] propose a policy based approach for (supervoxel,feature) selection in a budgeted semantic video labeling. Our work is dif-



Stage 1: Running inference policy for T steps.



Stage 2: Predicted pixel labels are propagated to remaining $M - T$ unlabeled frames.

Figure 1: Two stages of our approach. Given a video with M frames and time budget B, in Stage 1, LSTM-based policy sequentially selects a subset of frames for segmentation by a CNN, $\boldsymbol{f}$. In Stage 2, the missing pixel labels are interpolated by a one-layer convolution filter, $\boldsymbol{g}$, using neighboring semantic segmentations.

ferent from theirs in : 1) Unlike their approach, which is based on CRF based supervoxel labeling along the video, we use CNN based frame-level model aumgneted with a label propagation network. 2) Instead of the classication-based approximate policy iteration, we use a recurrent policy gradient method to learn the budget-aware policy.

3. Problem Formulation

Given a video, $\boldsymbol{x}$, with M frames and budget B, our goal is to accurately assign a label to each pixel in $\boldsymbol{x}$ in time less than B. Let $\boldsymbol{f}$ be a *segmentation function* that takes the pixels $\boldsymbol{x}_i$ of frame i and returns a semantic segmentation of $\boldsymbol{x}_i$ using time $c_{\boldsymbol{f}}$. In particular, $\boldsymbol{f}(\boldsymbol{x}_i)$ gives a posterior distribution over class labels for each individual pixel. We specify $\boldsymbol{f}$ as a CNN following two prior approaches [2, 23]. To meet the budget constraint, we apply $\boldsymbol{f}$ to only a subset of frames and interpolating the resulting segmentations to other frames. For this we use a *visual attention policy*, π, which sequentially selects frames for labeling by $\boldsymbol{f}$ (details in Section 4.1). Given the output of $\boldsymbol{f}$ at the selected frames, remaining frames are labeled by an *interpolation function*, $\boldsymbol{g}$, which uses nearby outputs of $\boldsymbol{f}$ to interpolate semantic segmentations to yet unprocessed frames. Importantly, the time cost of applying $\boldsymbol{g}$, $c_{\boldsymbol{g}}$, is significantly smaller than the time cost of applying $\boldsymbol{f}$, $c_{\boldsymbol{f}}$, which allows for time savings compared to labeling all frames via $\boldsymbol{f}$.

Since the goal is to produce a segmentation within a time budget B, we must decide when to stop selecting frames with π in order to meet the budget constraint. For this purpose, we can divide the total time required to compute an output into two components. The first time component is the time required to apply the policy π for T steps and also apply $\boldsymbol{f}$ at the selected frames. Let $\boldsymbol{u}^{(T)} \in \{0,1\}^M$ denote an indicator vector over video frames, where $u_i^{(T)} = 1$ means that frame i has been segmented by π, and $u_i^{(T)} = 0$, otherwise. Also let $U^{(T)} = |\boldsymbol{u}^{(T)}| \leq T$ be the number of distinct frames selected by π, noting that this can be less that T if π happens to select a frame twice.[1] The second time component involves applying $\boldsymbol{g}$ to frames with zero values in $\boldsymbol{u}^{(T)}$. If we let c_π denote the cost of applying the policy, then the total runtime is given by:

$$\begin{aligned} C(T) &= c_\pi T + c_{\boldsymbol{f}} U^{(T)} + c_{\boldsymbol{g}}(M - U^{(T)}) \\ &= c_\pi T + (c_{\boldsymbol{f}} - c_{\boldsymbol{g}}) U^{(T)} + c_{\boldsymbol{g}} M \end{aligned} \quad (1)$$

Using this runtime formula it is easy determine when to stop the policy. After T policy selections, we must stop before running step $T+1$ if this could result in $C(T+1) > B$.

In our work, π, $\boldsymbol{g}$, and $\boldsymbol{f}$ will be represented via deep neural networks. Since $\boldsymbol{f}$ is a pre-trained model, model parameters consist of the parameters of π and $\boldsymbol{g}$, i.e. $\theta = (\theta_\pi, \theta_{\boldsymbol{g}})$. Given a video $\boldsymbol{x}$ and time budget B, we let $\hat{\boldsymbol{y}}(\boldsymbol{x}, B, \theta)$ denote the output of the above semantic segmentation process using parameters θ applied to $\boldsymbol{x}$ with budget B. When clear from context we will denote this via $\hat{\boldsymbol{y}}$. Note that $\hat{\boldsymbol{y}}_i$ gives the posterior probability over labels for each pixel in frame i of the video. The loss of a frame labeling will be denoted by $\Delta(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i)$, where $\boldsymbol{y}_i$ is the ground truth labeling for frame i and Δ is an average cross-entropy loss for pixels in frame i. Further, the loss over M video frames is defines as:

$$L_\theta(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{M} \sum_{i=1}^{M} \Delta(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i). \quad (2)$$

[1] While π will rarely reselect a frame, we cannot rule out this possibility since π is learned and will have some imperfections.

Given a training set of N labeled videos $\{(\boldsymbol{x}^n, \boldsymbol{y}^n\}$ the goal of learning is to find θ that minimizes the following

$$\theta^* = \arg\min_{\theta} \left[L(\theta) = \mathbb{E}(L_\theta) \approx \frac{1}{N} \sum_{n=1}^{N} L_\theta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n) \right] \quad (3)$$

Unfortunately, even for simple choices for π, $\boldsymbol{f}$, and $\boldsymbol{g}$, the gradient of $L(\theta)$ does not have a closed form, due to the sequential nature of the process used to construct $\hat{\boldsymbol{y}}^n$.

4. Learning and Representation

In this section, we describe our representation for π and $\boldsymbol{g}$, noting that we use existing image segmentation models for $\boldsymbol{f}$, SegNet [2] and BayesianSegNet [23], whose detailed specification can be found in the respective references. We then describe our approach for jointly learning these functions by drawing on policy-gradient estimation techniques.

4.1. LSTM-based Policy

Our temporal attention policy makes a sequence of frame selections based on the local information perceived around the most recently selected frame, which we will refer to as the current frame. Note that the local observation input to the policy at each step only captures part of the global state of the inference process, s_t. This choice to limit the observations to only a local window around the current frame is motivated by the desire to allow for π consider long videos of different length and to make fast decisions, since otherwise the intelligent selection would be too costly to pay off.

However, limiting the size of local observation window can lead to sub-optimal decisions if optimal decisions depend on a wider context. To help address this we represent π using a recurrent neural network – namely, an LSTM, which attempts to learn a hidden state that captures the more global context of the inference process. Due to its ability to memorize information from previous decisions made by π, the LSTM has been shown to successfully model problems with non-Markovian state transitions such as ours and have a number of recent empirical successes in sequential decision making [34, 16, 47].

In particular, when the current frame at time t is i, the LSTM-based policy π makes a decision based on:

1) The local information in a video neighborhood $\mathcal{N}_i$ centered around i. This is captured in an observation vector $o_t = [\boldsymbol{z}_{\mathcal{N}_i}, \phi(\mathcal{N}_i), l_t]$, where $\boldsymbol{z}_{\mathcal{N}_i}$ is an indicator vector that indicates whether each frame in $\mathcal{N}_i$ has been previously selected and processed by $\boldsymbol{f}$, $\phi(\mathcal{N}_j)$ is the average of per-class confidences predicted by $\boldsymbol{f}$ in $\mathcal{N}_i$ (yielding one input image per class), and $l_t \in [0, 1]$ is the normalized location of the current frame at time t (e.g. the middle frame has value 0.5). The inclusion of l_t was helpful in encouraging the policy to cover the entire video extent. The input to the LSTM π at step t is o_t.

2) The LSTM's hidden variables $\boldsymbol{h}_{t-1}$, which summarizes the previous observations up to time t.

To summarize, the global state at time t is approximated by the internal state of the LSTM, $\boldsymbol{h}_t$, which depends on the current observation o_t and the previous state $\boldsymbol{h}_{t-1}$. Given h_t, the output of $\pi(h_t)$ is the location of the next observation $l_{t+1} \in [0, 1]$. Note that our formulation allows the policy to perform jumps forward and backward in time. Note that π is defined as a probabilistic policy, which is a convenient choice for our policy-gradient learning algorithm defined later. To improve exploration at training time, instead of using l_{t+1}, the next location is sampled from a Gausssian distribution with a mean equal to l_{t+1} and a fixed variance.

4.2. Interpolation Model

The goal of $\boldsymbol{g}$ is to estimate the pixel labels of frames not selected by π. Efficient and reliable interpolation of labels has been demonstrated on videos with smooth motions, i.e., strong correlations between neighboring frames [6, 7, 40]. Intuitively, given the posterior of class labels for frames j in the neighborhood of frame i, and the amount of change observed between frames i and j, our convolution filter $\boldsymbol{g}$ is learned to estimate labels of pixels in i, i.e. to output a posterior distribution over class labels for each pixel in i. The input to $\boldsymbol{g}$ is defined as an ordered set of pixel label predictions for the closest labeled frames j on two sides of frame i along with the additional channels containing pixel-wise frame differences between frames j and i. In our experiments, $\boldsymbol{g}$ is defined as a single layer convolution filter of size 5×5 with $2 \cdot (\#\text{ classes} + 1)$ input channels.

4.3. Joint Learning of the Parameters

The goal is to jointly learn the parameters of π and $\boldsymbol{g}$ ($\theta = \{\theta_\pi, \theta_{\boldsymbol{g}}\}$) by minimizing the labeling loss of a sequence of policy actions, taken from the initial state s_0 when no frames are selected until s_T, when the total runtime $C(T) \leq B \leq C(T+1)$.

Recall that $\hat{\boldsymbol{y}}_i^n$ is the estimated output for frame i in video $\boldsymbol{x}^n$. Let $\boldsymbol{u}^{(t)}$ be the indicator vector showing the selected frames in the entire video after running the policy for t steps. We can formally define the estimated output at each time step t as:

$$\hat{\boldsymbol{y}}_i^n(t) = \left[\boldsymbol{u}_{ni}^{(t)} \boldsymbol{f}(\boldsymbol{x}_i^n) + (1 - \boldsymbol{u}_{ni}^{(t)}) \boldsymbol{g}(\boldsymbol{x}^n)) \right] \quad (4)$$

The main difficulty is that the estimated output $\hat{\boldsymbol{y}}$ for the entire video, is computed through a sequence of decisions made by the policy which results in a non-decomposable, non-differentiable objective function. The decisions that the policy makes at any time depends on a history of decisions that the policy made in previous time steps and influences

the decisions available to the policy in the future. This is a long-standing problem in the study of reinforcement learning algorithms. To address this problem, the REINFORCE algorithm [44] and the recurrent policy gradient approach [43] approximate the gradients of the non-decomposable objective function which helps to efficiently learn the policy using stochastic gradient descent.

To follow the general reinforcement learning formulation, let r_t be the immediate reward associated with state s_t. Since $s_t \approx \boldsymbol{h}_t$ we define r_t as :

$$r_t(\boldsymbol{h}_t) = L_\theta(\boldsymbol{y}, \hat{\boldsymbol{y}}(t)) - L_\theta(\boldsymbol{y}, \hat{\boldsymbol{y}}(t-1)), \tag{5}$$

where L_θ is the labeling loss for the video defined in eq. (2). Intuitively, eq. (5) states that the policy earns an immediate reward equal to the decrease in labeling error achieved by selecting a frame (or pay a penalty if the labeling error increases). Let $R_t(H_t)$ be the discounted accumulated reward starting from state s_t and continuing the policy up to final state, s_T:

$$R_t(H_t) = \sum_{t'=t}^{T} \lambda^{t-t'} r_t(\boldsymbol{h}_t), \tag{6}$$

where $\lambda \in (0, 1)$ is the discount factor and $H_t = \{\boldsymbol{h}_t, \boldsymbol{h}_{t+1}, ..., \boldsymbol{h}_T\}$ represents a history of LSTM's hidden variables. H_0 can be interpreted as the trajectory of observations for a sample run of the policy from the initial state. For simplification purposes we use H for H_0 and R for R_0 in the rest of this paper. The goal is to find the parameters θ^*, that maximizes $J(\theta)$ redefined as:

$$J(\theta) = E[R(H)] = \int p(H|\theta) R_\theta(H) dH, \tag{7}$$

where $p(H|\theta)$ is the probability of observing a sequence of hidden states H, given a policy defined by parameters θ. It is easy to show that minimizing $L(\theta)$ in eq. (3) is equivalent to maximizing $J(\theta)$ in eq. (7). Let θ_π and $\theta_{\boldsymbol{g}}$ define the gradient of the objective function $J(\theta)$ in eq. (7) with respect to the LSTM policy and interpolation networks parameters. Although it is possible to jointly learn the parameters of the LSTM policy and the interpolation networks, given the stochastic nature of the policy, in practice we observed that the iterative approach works better. Alg .1 shows our proposed training procedure:

Computing $\bigtriangledown_{\theta_\pi} J$: The gradient with respect to the policy parameters is given by:

$$\bigtriangledown_{\theta_\pi} J = \int \big[\nabla_{\theta_\pi} p(H|\theta) R_\theta(H) + p(H|\theta) \nabla_{\theta_\pi} R_\theta(H)\big] dH \tag{8}$$

Note that given the hidden state sequence H, which determines the history of selected frames, the reward function does not depend on the policy parameters, yielding

Algorithm 1 Training procedure for our Budget-Aware semantic segmentation model

Input: N Training videos
Output: Learned parameters $\{\theta_\pi, \theta_{\boldsymbol{g}}\}$.
1: pre-train the interpolation network, $\boldsymbol{g}$ % note: training examples are generated from uniform sampling of frames in each video.
2: initialize the policy parameters, θ_π
3: **for** number of iterations **do**
% generate trajectories to train π
4: $\{H^n\}_{n=1}^N \leftarrow$ apply π for T steps
% interpolate labels for each video
5: $\{\hat{\boldsymbol{y}}^n\}_{n=1}^N \leftarrow$ using eq. (4)
% Update policy parameters:
6: $\theta_\pi \leftarrow - \bigtriangledown_{\theta_\pi} J$
% generate trajectories to train $\boldsymbol{g}$
7: $\{H^n\}_{n=1}^N \leftarrow$ aplly π for T steps
% Update interpolation parameters:
8: $\theta_{\boldsymbol{g}} \leftarrow - \bigtriangledown_{\theta_{\boldsymbol{g}}} J$
9: **end for**

$\bigtriangledown_{\theta_\pi} R_\theta(H) = 0$. To further simplify (8) we need to define $\bigtriangledown_{\theta_\pi} p(H|\theta)$. Note that $p(H|\theta)$ can be factorized as $p(H|\theta) = p(\boldsymbol{h}_0) \prod_{t=1}^{T} p(\boldsymbol{h}_t|\boldsymbol{h}_{t-1}) \pi(l_t|\boldsymbol{h}_{t-1}, o_t)$, where $o_t = [\boldsymbol{z}_{\mathcal{N}_j}^{(t)}, \phi(\boldsymbol{f}_{\mathcal{N}_j}^{(t)}), l_t]$ and the same notation π is used to denote the last softmax layer of the LSTM. From the above we have $\log p(H|\theta) = const + \sum_{t=0}^{T} \log \pi(l_t|\boldsymbol{h}_{t-1}, o_t)$ which results in the following gradient:

$$\bigtriangledown_{\theta_\pi} \log p(H|\theta) = \sum_{t=0}^{T} \nabla_\theta \log \pi(l_t|\boldsymbol{h}_{t-1}, o_t)$$

Monte Carlo integration is used to approximate the integration over the probability of observing a sequence of hidden states. Particularly the approximate gradient is computed by running the current policy on N given videos to generate N trajectories which result in the following approximate gradient:

$$\bigtriangledown_{\theta_\pi} J \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T} \big[\nabla_{\theta_\pi} \log \pi(l_t^n|\boldsymbol{h}_{t-1}^n, o_t^n) R_t(\boldsymbol{h}_t^n)\big]. \tag{9}$$

Policy gradient approaches suffer from the high variance of the gradient estimates. Following the common practice [44] we subtract a bias from the expected reward, R. Instead of a constant bias, we set the bias value to be the reward obtained following a random jump policy.

Computing $\bigtriangledown_{\theta_{\boldsymbol{g}}} J$: Analogous to eq. (8), the gradient with respect to the interpolation parameters is defined as $\bigtriangledown_{\theta_{\boldsymbol{g}}} J = \int \big[\nabla_{\theta_{\boldsymbol{g}}} p(H|\theta) R_\theta(H) + p(H|\theta) \nabla_{\theta_{\boldsymbol{g}}} R_\theta(H)\big] dH$. Note that since $\boldsymbol{g}$ is applied after frames are selected, the

hidden state probability does not depend on the interpolation function, i.e. $\bigtriangledown_{\theta_g} p(H|\theta) = 0$ which results in:

$$\bigtriangledown_{\theta_g} J = \int p(H|\theta) \bigtriangledown_{\theta_g} R_\theta(H) dH. \tag{10}$$

Recall that Δ is the average cross-entropy loss for pixels in frame i. It is easy to derive the following gradient:

$$\bigtriangledown_{\theta_g} L_\theta(\boldsymbol{y}, \hat{\boldsymbol{y}}(t)) = \frac{1}{|\boldsymbol{u}^{(t)}|} \sum_{\{i|u_i^{(t)}=0\}} \bigtriangledown_{\theta_{\boldsymbol{g}}} \Delta(\boldsymbol{y}_i, \boldsymbol{g}(\boldsymbol{x})). \tag{11}$$

Intuitively the labeling error of the frames which have not been selected by the policy is considered in computing the gradient with respect to the parameters of $\boldsymbol{g}$. Given a video $\boldsymbol{x}$ and applying (5), (7), and (11), it is easy to derive the following:

$$\bigtriangledown_{\theta_g} R_\theta(H) = \sum_{t=0}^{T} \lambda^t \bigtriangledown_{\theta_g} [L_\theta(\boldsymbol{y}, \hat{\boldsymbol{y}}(t)) - L_\theta(\boldsymbol{y}, \hat{\boldsymbol{y}}(t-1))] \tag{12}$$

Using the same Monte Carlo integration technique we derive the following approximate gradient:

$$\bigtriangledown_{\theta_{\boldsymbol{g}}} J \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=0}^{T} \lambda^t \bigtriangledown_{\theta_g} [L_\theta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n(t)) - L_\theta(\boldsymbol{y}^n, \hat{\boldsymbol{y}}^n(t-1))] \tag{13}$$

5. Results

We use the following datasets: 1) CamVid [5], and 2) KITTI [15]. Both datasets are recorded in uncontrolled environment, and present challenges in terms of occlusions, and variations of motions, shapes, and lighting.

Implementation: The LSTM model contains two hidden layers of 1024 hidden units . For training, we generate sequences of continuous frames per each training video where the sequence size is set to be 90 frames in both datasets. Also since the video datasets for semantic segmentation do not provide ground truth for all frames (e.g. CamVid provides labels for every 30 frames), for training, the output of $\boldsymbol{f}$ is considered as ground truth for frames without human annotated labels. Note that since the goal is to perform as well as a model that uses $\boldsymbol{f}$ across the entire video, it is reasonable to consider outputs of $\boldsymbol{f}$ as the ground truth. In our evaluation, the budget is defined in terms of the percentage of the maximum required budget needed to apply $\boldsymbol{f}$ for all video frames. Following [14], to improve convergence properties, we start with $\lambda = 0.9$ and gradually update it after each epoch using: $\lambda_{e+1} = 1 - 0.98(1 - \lambda_e)$.

We implement our interpolation and policy modules in tensorflow[2] and use the publicly available code for implementations of $\boldsymbol{f}$. Experiments are performed on an Intel quad core-i7 CPU and 16GB RAM on a single Tesla k80.

Variations of our approach and the baselines: We define two variants for the policy: i) REG: deterministically selects T frames uniformly starting from the first frame, ii) LSTM: learns the proposed model in Sec. 4.1. For interpolation we define the following variants: i) OPT: is the baseline approach proposed in [40] which uses dense optical flow to track the points from both forward and backward directions and then propagates the labels from the closest labeled points. More sophisticated label propagation approaches [30, 12, 13] are more expensive and are not suitable as a low-cost interpolation baseline. ii) CNN: learns the proposed interpolation filter in Sec. 4.2. The combination of [π = LSTM:$\boldsymbol{g}$ = CNN] is our approach and all other combinations are considered as baselines. We evaluate the validity of our approach on two deep semantic image segmentation models, i) SegNet[2] and ii) BayesianSegNet[23]. We would like to reemphasize that, $\boldsymbol{f}$ is considered as a black-box in our framework. While authors in [48, 11] reported higher accuracy compared to [2, 23], the above two models are only chosen because of the large variation in their accuracies and processing times which allows us to explore the generalizability of our framework in various settings.

5.1. Results on CamVid

The CamVid dataset has five videos of road scenes from a moving camera of length up to 6120 frames. Following prior work, we focus on the 11 most common object class labels. Ground-truth labels are available at every 30 frames. We use the standard test-train split as in [2] and similarly resize the frames to 360×480 pixels. The average per frame inference time is 165 ms for SegNet and 1450 ms for BayesianSegNet.

Table 1 shows the results for three different budgets and $\mathcal{N} = 7$. For $B = 0.1 \cdot B_{\text{max}}$ and $B = 0.25 \cdot B_{\text{max}}$ our [LSTM:CNN] outperforms all other variants in both class average accuracy and mean intersection over union [3]. One interesting observation is the contribution of each module in the accuracy. Based on the result, although both LSTM and CNN provides slight improvement in accuracy when applied independent of each other, they improve the accuracy with a larger margin when learned together. For $B = 0.5 \cdot B_{\text{max}}$, we observe a different pattern. Although our [LSTM:CNN] provides a better result, considering the relative accuracy between [LSTM:OPT] and [REG:OPT] and the same comparison between [REG:CNN] and [REG:OPT] it seems that accuracy boost is mostly due to the interpolation model.

[2] https://www.tensorflow.org/

[3] Intersection over Union (I/U) for one class = $\frac{tp}{tp+fp+fn}$

B	Method	Road	Building	Sky	Tree	Side-Walk	Car	Column-Pole	Fence	Pedestrian	Bicycle	Sign	Class Avg	Mean I/U
$\boldsymbol{f}$ = SegNet [2]														
$B_{\max}$	All Frames	88.0	87.3	92.3	80.0	29.5	97.6	57.2	49.4	27.8	84.8	30.7	65.9	50.2
$0.1 \cdot B_{\max}$	[REG:OPT]	56.4	46.4	53.7	42.7	10.3	60.3	29.3	18.5	9.5	36.8	10.7	34.1	26.7
	[REG:CNN]	57.2	47.0	51.5	43.8	11.2	60.9	28.4	19.3	9.8	37.9	11.1	34.4	26.9
	[LSTM:OPT]	60.8	50.3	54.9	51.2	16.8	70.1	32.7	25.6	15.7	44.2	14.7	39.7	30.2
	[LSTM:CNN]	63.4	58.8	61.9	54.9	17.3	73.9	38.1	31.3	19.5	51.7	20.8	44.7	33.8
$0.25 \cdot B_{\max}$	[REG:OPT]	70.4	69.1	73.2	63.8	23.6	81.9	45.4	39.1	20.9	65.9	23.8	52.5	40.2
	[REG:CNN]	70.8	68.9	75.4	64.2	24.8	78.6	45.8	39.4	22.1	66.8	24.5	52.8	40.5
	[LSTM:OPT]	83.4	70.4	83.7	72.3	25.4	88.1	52.3	45.2	24.6	78.6	26.8	59.2	45.4
	[LSTM:CNN]	81.1	80.3	82.9	73.7	27.6	89.8	54.2	45.9	25.8	78.0	28.4	60.7	46.2
$0.5 \cdot B_{\max}$	[REG:OPT]	83.2	82.6	86.3	75.4	27.5	93.2	53.7	46.8	25.9	80.1	29.1	62.2	47.0
	[REG:CNN]	81.6	82.1	86.0	75.7	27.4	90.8	53.1	46.9	25.8	81.4	27.6	61.7	46.9
	[LSTM:OPT]	81.5	82.0	86.7	75.8	27.1	91.4	52.9	46.4	25.2	80.6	27.5	61.6	46.7
	[LSTM:CNN]	84.1	83.7	87.4	76.2	27.9	93.4	54.3	46.3	26.3	80.5	29.3	62.7	47.7
$\boldsymbol{f}$ = Bayesian SegNet [23]														
$B_{\max}$	All Frames	80.4	85.5	90.1	86.4	67.9	93.8	73.8	64.5	50.8	91.7	54.6	76.3	63.1
$0.1 \cdot B_{\max}$	[REG:OPT]	51.6	49.3	59.6	41.9	44.7	46.9	34.4	35.8	24.5	45.2	20.7	41.3	34.8
	[REG:CNN]	52.7	52.8	57.8	44.7	46.2	50.3	36.8	36.9	25.2	46.7	21.6	42.9	36.1
	[LSTM:OPT]	59.2	57.8	63.2	54.9	49.1	57.6	42.4	40.1	31.3	56.7	30.5	49.3	40.8
	[LSTM:CNN]	60.3	60.1	64.8	56.7	50.3	60.1	46.8	42.3	33.7	59.4	31.6	51.5	42.3
$0.25 \cdot B_{\max}$	[REG:OPT]	60.8	65.8	70.1	66.3	51.6	70.8	57.1	49.7	38.3	70.9	41.5	58.4	47.9
	[REG:CNN]	62.5	65.2	71.5	68.4	52.3	71.2	60.3	52.2	39.9	70.4	41.9	59.6	49.3
	[LSTM:OPT]	76.3	81.1	83.8	81.4	63.1	87.5	68.9	60.9	47.3	85.1	51.3	71.5	58.7
	[LSTM:CNN]	76.0	80.9	85.7	82.8	64.6	88.1	70.4	61.2	48.8	84.6	52.5	72.3	59.4
$0.5 \cdot B_{\max}$	[REG:OPT]	75.7	80.4	83.8	82.1	64.3	89.5	69.5	60.2	48.2	86.3	51.3	71.9	59.0
	[REG:CNN]	75.8	80.9	86.4	82.5	64.8	89.8	69.5	61.1	48.5	87.5	51.8	72.6	59.6
	[LSTM:OPT]	75.2	80.5	85.6	82.4	63.9	89.0	70.2	60.1	47.9	85.8	51.1	72.0	58.8
	[LSTM:CNN]	77.1	81.9	86.2	81.7	65.1	88.7	69.3	61.8	49.1	88.2	52.8	72.9	59.8

Table 1: Comparison with different variations of our budget-aware inference on CamVid. The budget is defined as a fraction of the maximum budget required to run the original methods, [2, 23] for each frame, denoted as $B_{\max}$

Method	Time for π	Time for $\boldsymbol{g}$	Time for $\boldsymbol{f}$	Class-Avg
$\boldsymbol{f}$ = SegNet [2]				
REG+CNN($0.25 \cdot B_{\max}$)	0	130.4	251.3	53.5
REG+CNN($0.5 \cdot B_{\max}$)	0	90.6	567.2	61.7
LSTM+CNN($0.25 \cdot B_{\max}$)	23.1	125.8	256.7	61.7
LSTM+CNN($0.5 \cdot B_{\max}$)	5.2	92.1	441.6	62.8
$\boldsymbol{f}$= Bayesian SegNet [23]				
REG+CNN($0.25 \cdot B_{\max}$)	0	184.5	2928.4	62.3
REG+CNN($0.5 \cdot B_{\max}$)	0	97.3	4170.8	72.6
LSTM+CNN($0.25 \cdot B_{\max}$)	20.4	196.8	2934.7	72.3
LSTM+CNN($0.5 \cdot B_{\max}$)	8.8	113.9	4181.3	73.3

Table 2: Comparison of the processing times for two variations of our framework on a sample video from CamVid.

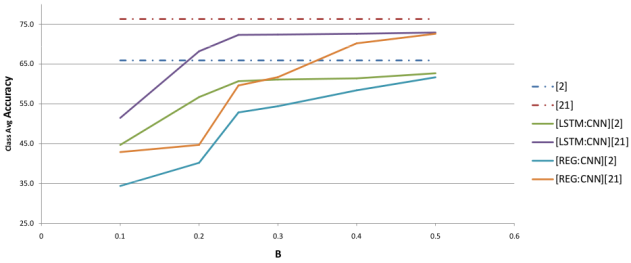


Figure 2: The class-average accuracy for different budget constraints using [1] =[23], [2] =[2] on CamVid.

Table 2 shows the processing time for policy execution, label interpolation, and semantic labeling using π, $\boldsymbol{g}$, and $\boldsymbol{f}$ for a sample video under different budget constraints. We observe that using $\boldsymbol{f}$ =[2], the policy selects almost 30% of the frames when $B = 0.25 \cdot B_{\max}$ and 50% of the frames when $0.5 \cdot B_{\max}$. For $\boldsymbol{f}$ =[23] which takes longer to run, the policy selects only 15% of the frames when $B = 0.25 \cdot B_{\max}$ and 40% of the frames when $0.5 \cdot B_{\max}$. This verifies our main hypothesis that the prior knowledge of the budget changes the behavior of the intelligent system and helps the intelligent agent to learn a particular frame selection pattern that improves the labeling accuracy for that budget.

Fig. 2 shows the class-average accuracy for different user-defined budgets. Despite a small accuracy drop, our approach keeps a consistent level of accuracy which shows the effectiveness of the learned frame selection and interpolation when $\frac{1}{4}B_{\max} < B < \frac{1}{2}B_{\max}$.

Qualitative results on the CamVid dataset are shown in Fig .3. The columns represent four consecutive frames of a sample video.

Results on KITTI: The KITTI dataset consists of videos from road scenes where eight class labels, (Building, Tree, Sign, Road, Fence, Pole, Sidewalk), are annotated in a subset of frames. Since the number of ground-truth annotations is much less than the number total of frames in videos. Instead of comparing with ground truth, for evaluating KITTI,

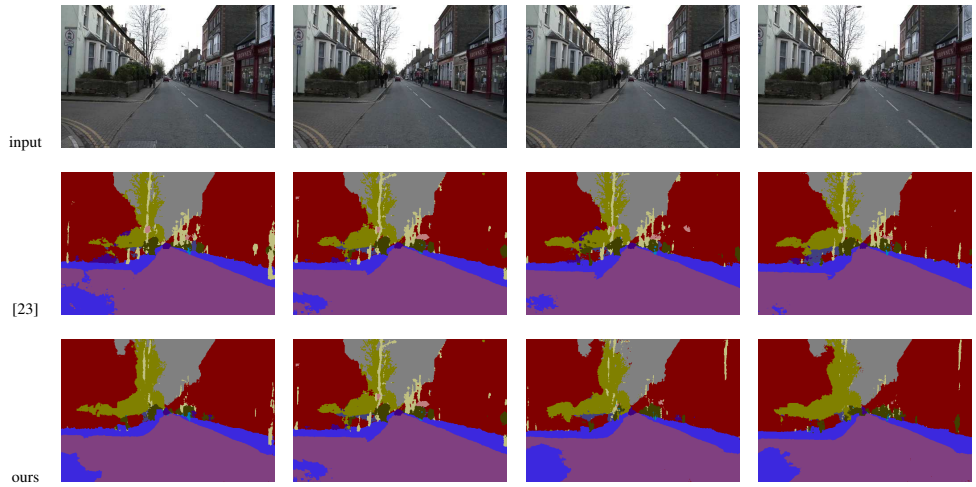


Figure 3: Frame samples from CamVid dataset. The top row shows the input and the middle row shows the result of applying [23] on individual frames. The bottom row shows the outputs of our LSTM:CNN approach. While only the frame in the second column is selected in our approach, qualitatively the results look very similar for other frames.

Budget	Method	Building	Tree	Sign	Road	Fence	Pole	Sidewalk	Class Avg
$0.25 \cdot B_{\text{max}}$	[REG:OPT]	89.7	81.2	74.2	83.3	61.5	72.2	85.3	78.2
	[REG:CNN]	90.5	82.3	77.4	86.9	67.3	74.7	86.8	80.8
	[LSTM:OPT]	91.2	86.8	80.2	90.2	71.2	78.4	89.2	83.9
	[LSTM:CNN]	91.9	92.1	89.5	93.5	78.3	83.2	90.2	88.4
$0.5 \cdot B_{\text{max}}$	[REG:OPT]	92.2	91.2	91.8	95.8	88.9	87.5	93.9	91.6
	[REG:CNN]	94.4	90.6	91.1	94.1	89.7	87.1	91.8	91.3
	[LSTM:OPT]	93.7	89.3	90.8	93.7	89.7	86.6	90.3	90.6
	[LSTM:CNN]	93.0	94.1	92.6	96.1	90.2	89.1	94.0	92.7

Table 3: Comparison of different variation of our budget-aware inference on KTTI. The budget is defined as a fraction of the maximum budget required to run the original method, [2] for each frame, denoted as B_{max}.

we compare our approach with a method which applies $\boldsymbol{f}$ on all video frames. Considering the fact that we are ultimately upper-bounded by the accuracy of the full run of $\boldsymbol{f}$ on the entire video, this is a reasonable evaluation. Note that during training we also use the very same outputs from applying $\boldsymbol{f}$ as the ground truth for training the policy and the interpolation network. We only re-train the last layer of the SegNet [2] to adjuect it for KITTI. Table 3 shows the per-class and class-average accuracy compared to results obtain from $\boldsymbol{f}$. For $B = 0.25 \cdot B_{\text{max}}$ our budget-aware inference achieves 88.4% accuracy and for $B = 0.5 \cdot B_{\text{max}}$ we achieve 90% accuracy. For a 4-fold speed up we have an accuracy reduction of only 11.5%.

6. Conclusion

We have addressed the problem of budgeted semantic video segmentation, where pixels of a video must be labeled within a time budget. We have specified a budget-aware inference for this problem that intelligently selects a subset of frames to run a deep CNN for semantic segmentation. Since CNN computation often dominates the cost of inference, our framework can provide substantial time savings in a principled manner. For selecting the subset of frames, we have formulated a visual-attention policy within the MDP framework, and used an LSTM as the policy model. We have also specified a new segmentation propagation function to label the unselected frames as a one-layer CNN. Our experiments show that our approach significantly improves on the accuracies of several strong baselines. The results also demonstrate that we can optimally adapt our method, from operating with no time bound to varying time budgets, such that it yields satisfactory performance for one-forth of the maximum budget, while maintaining an accuracy as close as possible to its performance for no time bound.

Acknowledgement

This work was supported in part by DARPA XAI, NSF RI1302700, and NSF IIS1619433.

References

[1] M. Amer, D. Xie, M. Zhao, S. Todorovic, and S.-C. Zhu. Cost-sensitive top-down/bottom-up inference for multiscale activity recognition. In *ECCV*, 2012.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[3] A. F. Behrooz Mahasseni, Sinisa Todorovic. Approximate policy iteration for budgeted semantic video segmentation. *CoRR*, abs/1607.07770, 2016.

[4] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11):1222–1239, 2001.

[5] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV*, 2008.

[6] I. Budvytis, V. Badrinarayanan, and R. Cipolla. Label propagation in complex video sequences using semi-supervised learning. In *BMVC*, 2010.

[7] A. Y. Chen and J. J. Corso. Propagating multi-class pixel labels throughout video frames. In *WNYIPW*, 2010.

[8] A. Y. Chen and J. J. Corso. Temporally consistent multi-class video-object segmentation with the video graph-shifts algorithm. In *WACV*, 2011.

[9] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[10] 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 preprint arXiv:1606.00915*, 2016.

[11] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[12] J. Fauqueur, G. Brostow, and R. Cipolla. Assisted video object labeling by joint tracking of regions and keypoints. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–7. IEEE, 2007.

[13] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015.

[14] V. François-Lavet, R. Fonteneau, and D. Ernst. How to discount deep reinforcement learning: Towards new dynamic strategies. *arXiv preprint arXiv:1512.02011*, 2015.

[15] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, pages 3354–3361. IEEE, 2012.

[16] K. Gregor, I. Danihelka, A. Graves, and D. Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[17] A. Grubb, D. Munoz, J. A. Bagnell, and M. Hebert. Speedmachines: Anytime structured prediction. *arXiv preprint arXiv:1312.0579*, 2013.

[18] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, pages 447–456, 2015.

[19] A. Jain, S. Chatterjee, and R. Vidal. Coarse-to-fine semantic video segmentation using supervoxel trees. In *ICCV*, 2013.

[20] A. Kae, B. Marlin, and E. Learned-Miller. The shape-time random field for semantic video labeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 272–279, 2014.

[21] J. Kappes, M. Speth, G. Reinelt, and C. Schnorr. Towards efficient and exact map-inference for large scale discrete computer vision problems via combinatorial optimization. In *CVPR*, 2013.

[22] S. Karayev, M. Fritz, and T. Darrell. Anytime recognition of objects and scenes. In *CVPR*, 2014.

[23] A. Kendall, V. Badrinarayanan, , and R. Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

[24] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. In *NIPS*, 2012.

[25] P. Lei and S. Todorovic. Recurrent temporal deep field for semantic video labeling. In *European Conference on Computer Vision*, pages 302–317. Springer, 2016.

[26] M. Liang, X. Hu, and B. Zhang. Convolutional neural networks with intra-layer recurrent connections for scene labeling. In *NIPS*, pages 937–945, 2015.

[27] B. Liu and X. He. Multiclass semantic video segmentation with object-level active inference. In *CVPR*, pages 4286–4294, 2015.

[28] B. Liu and X. He. Learning dynamic hierarchical models for anytime scene labeling. *ECCV*, 2016.

[29] B. Liu, X. He, and S. Gould. Multi-class semantic video segmentation with exemplar-based object reasoning. In *WACV*, 2015.

[30] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing: Label transfer via dense scene alignment. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1972–1979. IEEE, 2009.

[31] X. Liu, D. Tao, M. Song, Y. Ruan, C. Chen, and J. Bu. Weakly supervised multiclass video segmentation. In *CVPR*, 2014.

[32] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[33] O. Miksik, D. Munoz, J. A. Bagnell, and M. Hebert. Efficient temporal consistency for streaming video scene analysis. In *ICRA*, 2013.

[34] V. Mnih, N. Heess, A. Graves, et al. Recurrent models of visual attention. In *NIPS*, pages 2204–2212, 2014.

[35] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366, 2015.

[36] P. H. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene parsing. *arXiv preprint arXiv:1306.2795*, 2013.

[37] G. Roig, X. Boix, R. D. Nijs, S. Ramos, K. Kuhnlenz, and L. V. Gool. Active MAP inference in CRFs for efficient semantic segmentation. In *ICCV*, 2013.

[38] E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell. Clockwork convnets for video semantic segmentation. *ECCV*, 2016.

[39] B. Taylor, A. Ayvaci, A. Ravichandran, and S. Soatto. Semantic video segmentation from occlusion relations within a convex optimization framework. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 195–208. Springer, 2013.

[40] S. Vijayanarasimhan and K. Grauman. Active frame selection for label propagation in videos. In *European Conference on Computer Vision*, pages 496–509. Springer, 2012.

[41] D. Weiss, B. Sapp, and B. Taskar. Dynamic structured model selection. In *ICCV*, 2013.

[42] D. J. Weiss and B. Taskar. Learning adaptive value of information for structured prediction. In *NIPS*, 2013.

[43] D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber. Recurrent policy gradients. *Logic Journal of IGPL*, 18(5):620–634, 2010.

[44] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[45] C. Wojek and B. Schiele. A dynamic conditional random field model for joint labeling of object and scene classes. In *ECCV*, 2008.

[46] T. Wu and S.-C. Zhu. Learning near-optimal cost-sensitive decision policy for object detection. In *ICCV*, 2013.

[47] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. *arXiv preprint arXiv:1511.06984*, 2015.

[48] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[49] Y. Zhang and T. Chen. Efficient inference for fully-connected CRFs with stationarity. In *CVPR*, 2012.

[50] 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. *arXiv preprint arXiv:1502.03240*, 2015.