Predictive-Corrective Networks for Action Detection

Achal Dave Olga Russakovsky Deva Ramanan
Carnegie Mellon University

Abstract

While deep feature learning has revolutionized techniques for static-image understanding, the same does not quite hold for video processing. Architectures and optimization techniques used for video are largely based off those for static images, potentially underutilizing rich video information. In this work, we rethink both the underlying network architecture and the stochastic learning paradigm for temporal data. To do so, we draw inspiration from classic theory on linear dynamic systems for modeling time series. By extending such models to include nonlinear mappings, we derive a series of novel recurrent neural networks that sequentially make top-down **predictions** *about the future and then* **correct** *those predictions with bottom-up observations. Predictive-corrective networks have a number of desirable properties: (1) they can adaptively focus computation on "surprising" frames where predictions require large corrections, (2) they simplify learning in that only "residual-like" corrective terms need to be learned over time and (3) they naturally decorrelate an input data stream in a hierarchical fashion, producing a more reliable signal for learning at each layer of a network. We provide an extensive analysis of our lightweight and interpretable framework, and demonstrate that our model is competitive with the two-stream network on three challenging datasets without the need for computationally expensive optical flow.*

1. Introduction

Computer vision is undergoing a period of rapid progress. While the state-of-the-art in image recognition is disruptively increasing, the same does not quite hold for video analysis. Understanding human action in videos, for example, largely remains an unsolved, open problem. Despite a considerable amount of effort, CNN-based features do not yet significantly outperform their hand-designed counterparts for human action understanding [1, 48]. We believe that one reason is that many architectures and optimization techniques used for video have largely been inspired by those for static images (so-called "two-stream" models [38, 50, 51, 8]), though notable exceptions that directly process spatio-temporal volumes exist [42, 43]. In terms of benchmark results, two-stream models currently outperform the latter, perhaps because of the large computational demands of processing spatio-temporal volumes.

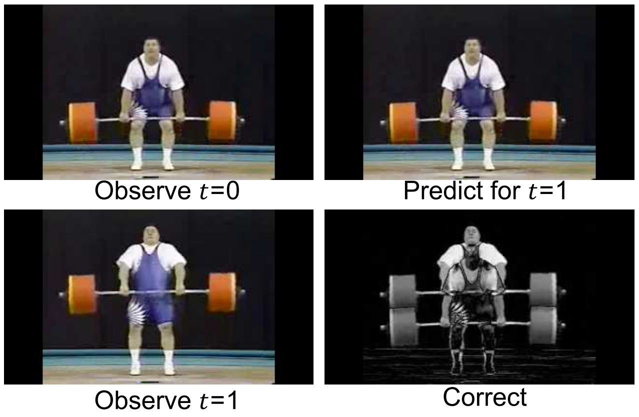

Figure 1. Our model first **predicts** the future and then updates its predictions with **corrections** from observing subsequent frames.

Recurrent models: An attractive solution to the above are *state-based* models that implicitly process large spatio-temporal volumes by maintaining a hidden state over time, rather than processing the whole block at once. Classic temporal models based on Hidden Markov Models (HMMs) or Kalman filters do exactly this. Their counterpart in the world of neural nets would be recurrent models. While an active area of research in the context of language [44, 6], they are relatively less explored for video-based feature learning (with the important exceptions of [56, 58, 40]). We posit that one reason could be the difficulty of stream-based training with existing SGD solvers. Temporal data streams are highly correlated, while most solvers rely heavily on uncorrelated *i.i.d.* data for efficient training [30]. Typical methods for ensuring uncorrelated data (such as random data permutations) would remove the very temporal structure that we are trying to exploit!

Our approach: We rethink both the underlying network architecture and stochastic learning paradigm. To do so, we draw inspiration from classic theory on linear dynamic systems for time-series learning models. By extending such iconic models to include nonlinear hierarchical mappings, we derive a series of novel recurrent neural networks that work by making top-down *predictions* about the future and *correct* those predictions with bottom-up observations (Fig. 1). Just as *encoder-decoder* architectures allow for neural networks to incorporate insights from clustering and sparse-coding [11], *predictive-corrective* architectures allow them to incorporate insights from time-series analysis: (1) adaptively focus computation on "surprising" frames where predictions require large corrections, (2) simplify learning in that only "residual-like" corrective terms need to be learned over time and (3) naturally decorrelate an input stream in a hierarchical fashion, producing a more reliable signal for learning at each layer of a network.

Prediction: From a biological perspective, we leverage the insight that the human vision system relies heavily on continuously predicting the future and then focusing on the unexpected [7, 25]. By utilizing the temporal continuity of video we are able to predict future frames in the spirit of [46, 47]. This serves two goals: (1) achieves consistency in predicted actions, reducing the chance that a single noisy frame-level prediction changes the model's interpretation of the video, and (2) results in a computationally efficient system, which is critical for real-world video analysis. If no significant changes are observed between frames then the computation burden can be significantly reduced.

Correction: Even more importantly, explicitly modeling appearance predictions allows the model to focus on correcting for novel or unexpected events in the video. In fine-grained temporal action localization, transitions between actions are commonly signified by only subtle appearance changes. By explicitly focusing on these residual changes our model is able to identify action transitions much more reliably. Further, from a statistical perspective, focusing on changes addresses a key challenge in learning from sequential data: it reduces correlations between consecutive samples, as illustrated in Fig. 2. While consecutive video frames are highly correlated (allowing us to make accurate predictions), *changes* between frames are not, increasing the diversity of samples observed during training.

Contributions: We introduce a lightweight, intuitive and interpretable model for temporal action localization in untrimmed videos. By making predictions about future frames and subsequently correcting its predictions, the model is able to achieve significant improvements in both recognition accuracy and computational efficiency. We demonstrate action localization results on three benchmarks: the standard 20 sports actions of THUMOS [15], the 65 fine-grained actions of MultiTHUMOS [55] and the 157 common everyday actions of Charades [37]. Our model is competitive with the two-stream network [38] on all three datasets without the need for computationally expensive optical flow. Further, it even (marginally) outperforms the state of the art MultiLSTM model on MultiTHUMOS [55].

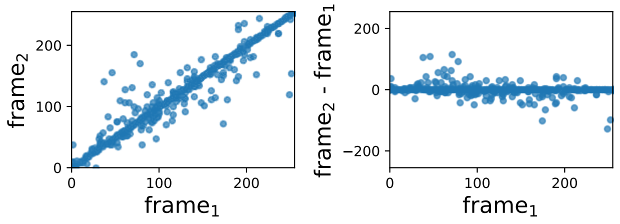

Figure 2. Every data point corresponds to a single location in two subsequent frames. The x-axis is the pixel intensity at this location in frame$_1$ and the y-axis is the pixel intensity at this location in frame$_2$ on the left and frame$_2$-frame$_1$ on the right. *(Left)* Consecutive video frames contain highly correlated information, allowing our model to make accurate and efficient *predictions* about future frames. *(Right)* Explicitly reasoning about frame differences removes correlation and allows the model to focus on reasoning about visual changes by making *corrections* to the predictions.

2. Related Work

Action recognition: There is a vast literature on action recognition from videos: to name a few, [19, 58] explore fusing image-based convolutional networks over time, [38] use RGB pixel information together with optical flow to capture motion, [14, 42, 43] extend image-based convolutional networks into 3D convolutional networks that operate on video "volumes" consisting of a fixed number of video frames. In contrast to these works, we focus on the more challenging task of temporal action localization.

Temporal action localization: A common way to extend action recognition models to temporal detection is through the sliding windows paradigm [48, 17, 49, 27, 57]. However, this is both computationally inefficient and prevents the model from leveraging memory over the video. Classical temporal models, on the other hand, can leverage information from the past as well as the future. These models generally rely on chain structured models that admit efficient inference, such as HMMs [33, 12] and CRFs [52]. More recent approaches for reasoning about memory generally focus on Recurrent Neural Networks (RNNs), which sequentially [55] or sporadically [56] process video frames and maintain an explicit memory of the previously observed frames. [20] develop a "Clockwork RNN" that maintains memory states that evolve at different speeds while processing a sequence; [32] extend this model to convolutional networks for semantic segmentation in videos. Our model follows similar intuition for temporal action detection.

Predictive models: It has been shown that leveraging global contextual information can be used to improve image [26] or video [23] understanding. Recent work has examined predicting the appearance and semantic content of future video frames [45, 46, 47, 54, 9, 22, 24]. Recently Srivastava et al. [40] train a recurrent neural network in an encoder-decoder fashion to predict future frames, and demonstrates that the learned video representation improves action recognition accuracy. However, to the best of our knowledge these insights have not been used for designing accurate end-to-end action localization models.

Accelerating learning: Recurrent neural nets are notoriously difficult to train because of the exploding gradients encountered during SGD [29]. We refer the reader to [3] for an excellent introduction to general SGD learning. Though naturally a sequential algorithm that processes one data example at a time, much recent work focuses on mini-batch methods that can exploit parallelism in GPU architectures or clusters [5]. One general theme is efficient online approximation of second-order methods [2], which can model correlations between input features. Batch normalization [13] computes correlation statistics between samples in a batch, speeding up convergence. Predictive-corrective networks naturally de-correlate batch statistics without needing expensive second-order computations (Fig. 2).

Interpretable models: Understanding the inner workings of models is important for diagnosing and correcting mistakes [28, 59]. However, despite some recent progress on this front [18], recurrent neural networks largely remain a mystery. By introducing a lightweight interpretable recurrent model we aim to gain some insight into the critical components of accurate and efficient video processing.

3. Predictive-Corrective Model

Consecutive video frames contain redundant information, which both causes needless extra computation and creates difficulty during training since subsequent samples are highly correlated. In our predictive-corrective model, we remove this redundancy by instead explicitly reasoning about changes between frames. This allows the model to focus on key visual changes, e.g., corresponding to human motion.

We first provide some intuition motivated by Kalman Filters. Then, we describe a procedural way to apply our model to image-based networks to create recurrent predictive-corrective structures for action detection. The model smoothly updates its memory over consecutive frames through residual corrections based on frame changes, yielding an accurate and efficient framework.

3.1. Linear Dynamic Systems

Setup: Consider a single-shot video sequence that evolves continuously over time. For a video frame at time t, let $\mathbf{x}_t$ denote the underlying semantic representation of the state. For example, on the standard THUMOS dataset [15] with 20 sports actions of interest, $\mathbf{x}_t$ could be a 20-dimensional binary vector indicating the presence or absence of each action within the frame. Instead of a discrete binary vector, we think of $\mathbf{x}_t$ as a smooth semantic representation: for example, actions can be decomposed into mini muscle motions and $\mathbf{x}_t$ can correspond to the extent each of these motions is occurring at time t. The action detection model is unaware of the underlying state $\mathbf{x}_t$ but is able to observe the pixel frame appearance $\mathbf{y}_t$ and is tasked with making an accurate semantic prediction $\hat{\mathbf{x}}_t$ of the state $\mathbf{x}_t$.

Dynamics: We model the video sequence as a linear dynamic system, which evolves according to

$$\begin{aligned} \mathbf{x}_t &= \mathbf{A}\mathbf{x}_{t-1} + \textit{noise} \\ \mathbf{y}_t &= \mathbf{C}\mathbf{x}_t + \textit{noise} \end{aligned} \tag{1}$$

In other words, the semantic state $\mathbf{x}_t$ is a noisy linear function of the semantic state at the previous time step $\mathbf{x}_{t-1}$, and the pixel-level frame appearance $\mathbf{y}_t$ is a noisy linear function of the underlying semantic action state $\mathbf{x}_t$. This is an imperfect assumption, but intuitively $\mathbf{x}_t$ can be thought of as action probabilities, $\mathbf{A}$ can correspond to the transition matrix between actions, and, if $\mathbf{x}_t$ is sufficiently high-dimensional, then a linear function can serve as a reasonable approximation of the appearance $\mathbf{y}_t$.

Kalman filter: Under this linear dynamic system assumption, the posterior estimate of the action state $\mathbf{x}_t$ is:

$$\hat{\mathbf{x}}_t = \underbrace{\hat{\mathbf{x}}_{t|t-1}}_{\text{prediction}} + \mathbf{K}(\underbrace{\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}}_{\text{correction}}) \tag{2}$$

where $\hat{\mathbf{x}}_{t|t-1}$ and $\hat{\mathbf{y}}_{t|t-1}$ are the prior prediction of $\mathbf{x}_t$ and $\mathbf{y}_t$ respectively given observations $\mathbf{y}_1 \ldots \mathbf{y}_{t-1}$ up to previous time steps $t-1$, and $\mathbf{K}$ is the Kalman gain matrix. We refer the reader to [41] for an overview of Kalman filters; for our purposes, we think of $\mathbf{K}$ as a learned non-linear function of the difference between actual and predicted frame appearance $\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}$. We analyze Eqn. 2 step by step.

State approximation: To make predictions of the semantic action space $\hat{\mathbf{x}}_{t|t-1}$ and of appearance $\hat{\mathbf{y}}_{t|t-1}$, we rely on the fact that the actions and pixel values of a video evolve *slowly* over time [53]. Using this fact, we can use the previous time step $t-1$ and approximate $\hat{\mathbf{x}}_{t|t-1} \approx \hat{\mathbf{x}}_{t-1}$ with our best prediction of the action state at the previous frame, intuitively saying that the transition matrix between actions in subsequent frames is near-identity. Further, we can assume that the video frame appearance is near constant and approximate $\hat{\mathbf{y}}_{t|t-1} \approx \mathbf{y}_{t-1}$ with the observed appearance of the previous frame. Eqn. 2 now simplifies to:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t-1} + g(\mathbf{y}_t - \mathbf{y}_{t-1}) \tag{3}$$

where g is a learned function, which helps compensate for the imperfect assumptions made here.

Learning: What remains is learning the non-linear function g from differences in frame appearance to differences in action state. We call this a *predictive-corrective block* and it forms the basis of our model described below.

3.2. Layered Predictive-Corrective Blocks

Setup: So far we described a general way to predict a hidden state $\mathbf{x}_t$ given observations $\mathbf{y}_t$. Instead of thinking

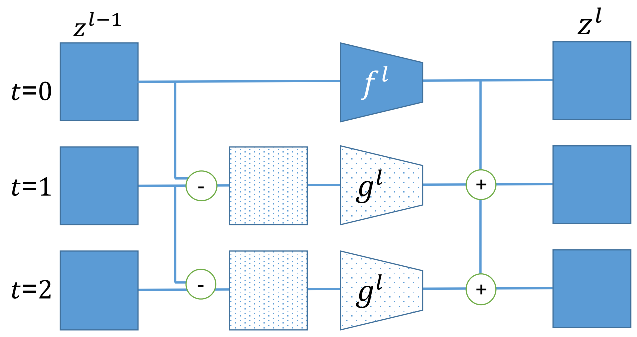

Figure 3. An instantiation of our predictive-corrective block. The filled and unfilled trapezoids correspond to f^i and g^l respectively.

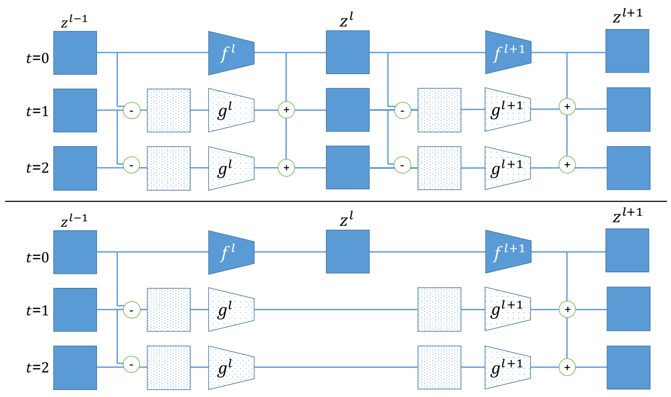

Figure 4. *(Top)* A predictive-corrective block placed at layer l and layer $l+1$. *(Bottom)* An equivalent but simplified network.

of $\mathbf{y}_t$ as the frame appearance and $\mathbf{x}_t$ as the semantic action state at time t, here we recursively extend our predictive models to capture the hierarchical layers of a deep network: we simply *model lower layers as observations that are used to infer the hidden state of higher layers*. Our model naturally combines hierarchical top-down prediction with hierarchical bottom-up processing prevalent in deep feedforward nets. Let us imagine that layers are computing successively more invariant representations of a video frame, such as activations of parts, objects, actions, etc. We use our model to infer latent parts from image observations, and then treat part activations as observations that can be used to infer objects, and so forth. Let $\mathbf{z}_t^l$ represent the latent activation vector in layer l at time t.

Single layer: Let us assume $\mathbf{z}_t^l$ evolves over time according to a linear dynamic system that generates observations in the layer below $\mathbf{z}^{l-1}$. Then the $\mathbf{z}_t^l$ can be predicted as $\hat{\mathbf{z}}_t^l$ by observing the temporal evolution of $\mathbf{z}^{l-1}$ using Eqn. 3:

$$\hat{\mathbf{z}}_t^l = \hat{\mathbf{z}}_{t-1}^l + g^l(\mathbf{z}_t^{l-1} - \mathbf{z}_{t-1}^{l-1}) \tag{4}$$

There are three things that deserve further discussion. First, the true latent state $\mathbf{z}_t^l$ can never be observed directly and we have to rely on predictions $\hat{\mathbf{z}}_t^l$. Second, the base case of the temporal recursion at time $t = 0$ needs to be considered. Third, the layer-specific function g^l needs to be learned to predict the evolution of layer l based on the evolution of layer $l-1$. We now address each of these in order.

Hierarchical model: The latent state of a layer $\mathbf{z}_t^l$ can never be observed except at the lowest layer $l = 0$ where $\mathbf{z}_t^0$ is the pixel appearance of the frame. Thus, at each time step t we initialize $\mathbf{z}_t^0$ with the pixel appearance, compute the predicted state $\hat{\mathbf{z}}_t^1$ using Eqn. 4, and use it as the observed $\mathbf{z}_t^1$ to compute $\hat{\mathbf{z}}_t^2$, continuing the layerwise recursion.

Temporal initialization: For the base case of the temporal recursion at time $t = 0$ we need a separate convolutional neural network f. This network does not consider the evolution of the dynamic system and can be thought of as a simple per-frame (action) recognition model. At the final layer L this network computes the action predictions $\hat{\mathbf{z}}_0^L = f(\mathbf{z}_0^0)$ from pixel activations $\mathbf{z}_0^0$; it can also be decomposed layerwise into $\mathbf{z}_0^l = f^l(\mathbf{z}_0^{l-1})$. In practice we train f jointly with g. Fig. 3 depicts an instantiation of the system for a single layer l, where f^l is used to process the initial frame and g^l is used to compute sequential updates.

Learning: Both the initial-frame function f and the residual function g need to be learned. At any time t, we know the pixel frame features $\mathbf{z}_t^0$ at the zeroth layer of the network and the desired action labels $\mathbf{z}_t^L$ at the final layer. To compute the action predictions at time $t = 0$ we use $\hat{\mathbf{z}}_0^L = f(\mathbf{z}_0^0)$. To compute the action predictions at time $t \neq 0$ we let $\Delta_t^l \triangleq \hat{\mathbf{z}}_t^l - \hat{\mathbf{z}}_{t-1}^l$ and rewrite the predictive-corrective block equations in Eqn. 4 as:

$$\Delta_t^L = g^L(\Delta_t^{L-1}) = g^L(g^{L-1}(\cdots g^1(\Delta_t^0))) = g(\mathbf{z}_t^0 - \mathbf{z}_{t-1}^0)$$

where $\mathbf{z}_t^0 - \mathbf{z}_{t-1}^0$ is the pixelwise difference between the current and previous frame. Now we can independently compute $\Delta_1^L, \ldots, \Delta_t^L$ using the network g and obtain desired action predictions $\hat{\mathbf{z}}_t^L = \hat{\mathbf{z}}_0^L + \sum_{i=1}^t \Delta_i^L$ for any time step t. The full system is depicted in Fig. 4. The known action labels $\mathbf{z}_t^L$ at every time step t provide the training signal for learning the networks f and g, and the entire system can be trained end-to-end.

3.2.1 Connections to prior art

Nonlinear Kalman filters: At this point, it is natural to compare our nonlinear dynamic model other nonlinear extensions of Kalman filtering [10]. Popular variants include the "extended" Kalman filter that repeatedly linearizes a nonlinear model [21], and the "unscented" Kalman filter that uses particle filtering to model uncertainty under known but nonlinear dynamic function [16]. Our work differs in that we assume simple linear dynamics (given by identity mappings), but model the data with complex (nonlinear) hierarchical observation models that are latently-learned from data without hierarchical supervision.

Recurrent networks: We also briefly examine the connection between our framework and a RNN formulation [56, 20, 55]. The update equation for RNN memory is $\mathbf{x}_t = \sigma(W\mathbf{y}_t + V\mathbf{x}_{t-1})$ for input $\mathbf{y}_t$, non-linearity σ and learned weights W and V. Similarly, our fully-connected predictive-corrective block in Eqn. 3 can be written as $\mathbf{x}_t = \mathbf{x}_{t-1} + \sigma(W(\mathbf{y}_t - \mathbf{y}_{t-1}))$. The key differences are (1) we use the past output $\mathbf{x}_{t-1}$ in a *linear* fashion, and (2) we maintain the previous input $\mathbf{y}_{t-1}$ as part of the memory. These imposed constraints are natural for video processing and allow for greater interpretability of the model. Concretely, our memory is simply the convolutional activations at the previous time step, and, thus is as interpretable as the activations of an image-based CNN (using e.g., [59]). Second, memory updates are transparent: we clear memory every few frames on re-initialization, and access it only to subtract it from the current convolutional activations, in contrast to the LSTM's more complex update mechanism.

3.3. Dynamic Computation

Coming back to our model, so far we discussed the case where layer activations evolve smoothly in a linear dynamic system. However, layer activations between subsequent frames may *not change* at all or may *change too much* to be modeled via smooth updates. Both cases are naturally incorporated into our predictive-corrective model, with the first case additionally yielding computational savings.

Static activations: Layer activations do not change at every time step within a video. This may be because the video depicts a static scene with no moving objects (e.g., in a surveillance camera) or because the frame rate is so high that occasionally subsequent frames appear identical. It may also be the case that while the low-level pixel appearance changes, the higher layers remain static (e.g., a "face" neuron that fires regardless of the face's pose or position). Within our model, this leads to $\Delta_t^l = 0$, eliminating the need for subsequent processing of the corrective block of this frame t for layers $l' > l$ and thus improving efficiency.

Shot changes: On the flip side, occasionally layer activations change so much between subsequent frames that a smooth update is not a reasonable approximation. Then we "re-initialize" by reverting to our initialization network f.

Dynamic updates: Concretely, let α_t^l be an indicator variable representing whether the change in all lower layers $l' < l$ is large enough to warrant corrective computation. Let δ_t^l be an indicator variable representing whether $\hat{\mathbf{z}}_t^l$ should be reinitialized, either because the change $|\hat{\mathbf{z}}_t^l - \hat{\mathbf{z}}_{t-1}^l|$ is too large or according to a preset layerwise clock rate [32, 20]. Then, we can rewrite Eqn. 4 as:

$$\hat{\mathbf{z}}_t^l = \begin{cases} \hat{\mathbf{z}}_{t-1}^l & \text{if } \alpha_t^l = 1 \\ f^l(\hat{\mathbf{z}}_t^{l-1}) & \text{if } \delta_t^i = 1 \\ \hat{\mathbf{z}}_{t-1}^l + g^l(\mathbf{z}_t^{l-1} - \mathbf{z}_{t-1}^{l-1}) & \text{else} \end{cases} \quad (5)$$

We analyze the effect of dynamic updates on both accuracy and efficiency in our experiments.

4. Experiments

We begin by presenting a detailed analysis of our model with experiments on a validation split of the MultiTHUMOS dataset [55] in Sec. 4.1. Leveraging this analysis, we then compare the optimal configuration of our predictive-corrective architecture with prior work in Sec. 4.2.

Implementation: For our initial and update models, we use the VGG-16 network architecture [39]. The model is initialized by training on ILSVRC 2016 [31]. We finetune the model on the per-frame action classification task for all actions, and use these finetuned weights to initialize both the initial and update networks in our model. All of our models are implemented using the Torch [4] deep learning framework. We will release source code, including hyperparameters and validation splits, for training and evaluating our models. For all of our experiments, we work with frames extracted from the videos at 10 frames per second. Each frame is resized to 256x256 pixels, and we take random crops of 224x224 for each frame.

4.1. Predictive-Corrective Model Analysis

To analyze the contributions from our proposed approach, we first compare a simple configuration of our approach to baseline models (Sec. 4.1.1). Next, we evaluate the trade-off of accuracy and efficiency in our framework (Sec. 4.1.2). Finally, we consider different model architectures by varying the placement of the predictive-corrective block in the VGG-16 architecture (Sec. 4.1.3).

Setup: MultiTHUMOS [55] contains 65 fine-grained action annotations on videos from the THUMOS 2014 dataset [15], which contain 2,765 trimmed training videos, 200 untrimmed training videos, and 213 untrimmed test videos. Of the 200 untrimmed training videos, we select 40 for validation, on which we report experiments below. We evaluate our predictions with per-frame[1] mean average precision (mAP). [55]

4.1.1 Comparison with baselines

Setup: We examine a simple variant of our model: the predictive-corrective block at the `fc7` layer, which uses frame-level corrections to update `fc7` activations. In this case, the initial function f and the update function g consist of the layers in VGG-16 up to `fc7`. Fig. 5 shows an instantiation of this with a reinitialization rate of 2. Here we consider the model with a reinitializion rate of 4 frames.

[1]Action detection accuracy may also be reported as mAP at specified intersection-over-union (IOU) thresholds, as in [34]. However, this requires post-processing predictions to generate action instances, and we choose not to do that as to not complicate our analysis.

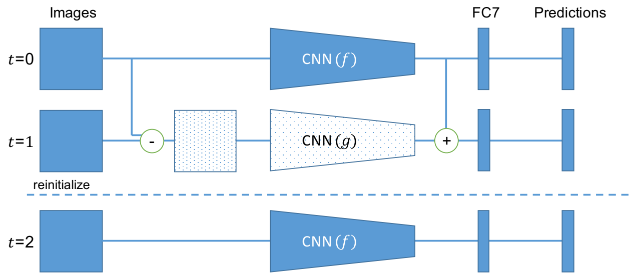

Figure 5. An instantiation of our predictive-corrective network: a predictive-corrective block between the input images and the `fc7` layer with a re-initialization rate of 2 frames.

Method	MultiTHUMOS mAP
Single-frame RGB	25.1
4-frame late fusion	25.3
Predictive-corrective (our)	**26.9**

Table 1. Our predictive-corrective model outperforms both baselines. (Per-frame mAP on MultiTHUMOS validation set.)

Baselines: We compare our model against the performance of baseline models that do not use our predictive-corrective blocks. To this end, we evaluate two models. First, we evaluate the single frame model finetuned to predict action labels for each frame of a video. Second, we consider a model similar to the *late fusion* model of [19] (or the *late pooling* model of [58]). It takes as input 4 frames (3 from previous time steps plus the current frame) and average pools their `fc7` activations before making a prediction of the action occurring at the current time step. When training this model we tie the weights corresponding to the three frames, which we found to empirically perform better than leaving all untied or tying all four branches together.

Results: Table 1 reports the results. These baselines explore the contribution of naive temporal information to our performance. While incorporating these cues provide a small 0.2% boosts over the baseline (25.1% mAP for single-frame vs 25.3% mAP for late fusion), it does not match the performance of our predictive-corrective model. Our model outperforms the single-frame model by 1.8% mAP: from 25.1% mAP of single-frame to 26.9% mAP for ours.

The single-frame model often relies mostly on the image context when making predictions, yielding many confident false positive predictions as shown in Fig. 6. For example, in the top row of Fig. 6 the single-frame model predicts the "clean and jerk" action based on the scene appearance even though the human is not currently performing the action. In contrast, our model is able to effectively use the predictive-corrective block to focus only on the moving parts of the scene and realize that the action is not yet being performed.

The precision-recall curves in Fig. 7 verify this intuition. The single-frame model consistently suffers from low precision as a result of making many false positive predictions.

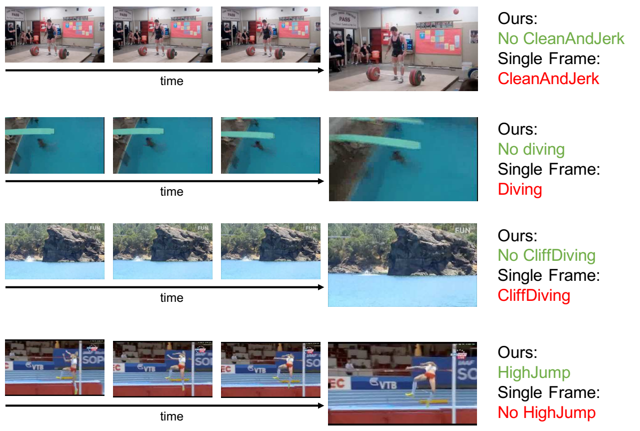

Figure 6. The single-frame model makes predictions based on the overall scene context whereas our predictive-corrective model leverages temporal information from 4 frames to focus on the scene changes and more accurately reason about actions.

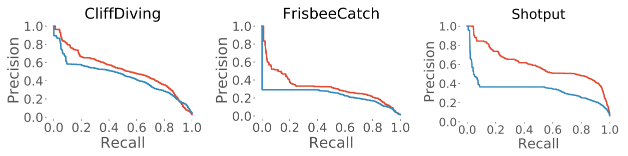

Figure 7. Precision-Recall curves for our model (orange) vs. single-frame (blue). The single-frame model often makes predictions based on context (e.g., "CliffDiving" in the presence of a cliff), leading to a lower precision than our model, which leverages temporal information to distinguish actions from context. (Per-frame precision/recall on MultiTHUMOS validation set.)

4.1.2 Test-time reinitialization

One advantage of our model is that it can be reinitialized dynamically at test time, as described in Sec. 3.3. We have seen results with varying training re-initialization rates in Table 3. However, these models can be applied in a different setting at test time. This can be useful, for example, if our training data contains videos with many shot changes, but we are interested in evaluating on smooth videos.

Static reinitialization: We experiment with different *train* and *test* reinitialization rates for our `fc7` predictive-corrective model in Table 2 for simplicity. The model trained to reinitialize every 4 frames can successfully reason about the video for up to 8 frames without reinitializing with only a modest drop in mAP, while the model trained on 8 frames can generalize to reasoning for up to 16 frames.

Dynamic reinitialization: In addition to static reinitialization rates, our model is able to *dynamically* decide when to re-initialize at test time. This allows it to use the corrective model when the video is evolving smoothly, and re-initialize only during big time changes. We implement this by thresholding the corrective term computed in the given frame; if its magnitude is greater than our threshold, we re-

Reinit	Train Reinit 4	Train Reinit 8
Test Reinit 2	26.9	25.9
Test Reinit 4	26.9	26.9
Test Reinit 8	25.4	**27.3**
Test Reinit 16	20.0	25.9

Table 2. Our model is able to reason about the video at test time for longer than it was trained for, with only modest losses in accuracy. (Per-frame mAP on MultiTHUMOS validation set.)

Configuration	mAP
`conv53` every 4	26.5
`fc7` every 4	26.9
`fc8` every 4	26.6
`conv33` every 1, `fc7` every 4	**27.2**
`conv43` every 2, `conv53` every 4	26.6
`conv53` every 2, `fc7` every 4	24.8

Table 3. Accuracy of different predictive-corrective architectures. (Per-frame mAP on MultiTHUMOS validation set.)

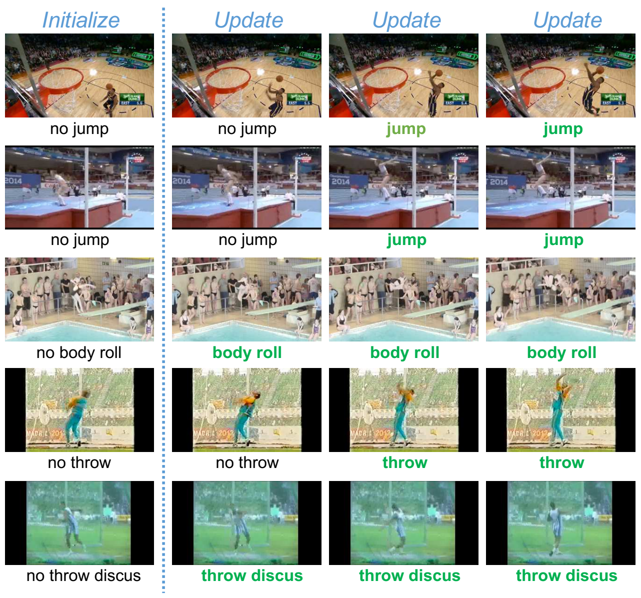

Figure 8. Qualitative results on the MultiTHUMOS validation set. Labels are our model's predictions for each frame. Our model initializes on the first frame and updates using the next three frames. Our update mechanism correctly recognizes the start of actions after initialization, and even corrects errors from initialization (last).

initialize the model. To avoid propagating mistakes for long sequences, we require the model to re-initialize at least every 4th frame. We find that by validating over a simple dynamic threshold on the norm of the correction, we can already achieve a small improvement in accuracy from the static reinitialization rate (27.2% mAP dynamic vs 26.9% mAP static). This suggests that using more advanced methods such as reinforcement learning to learn dynamic updates may yield further benefits in our frameworks.

Efficiency: Processing videos is computationally challenging due to the heavy redundancies between frames. Our model naturally allows us to avoid unnecessary computation on frames that are mostly redundant. We implement this by discarding frames when the corrective term is below a threshold. We find that we can dynamically discard nearly 50% of the frames, thus reducing the computational burden by a factor of two, while only slightly dropping performance (26.7% mAP with processing only half the frames vs 26.9% mAP with processing all frames). Note that this is not the same as randomly discarding frames, as our model still outputs predictions for *all* frames.

4.1.3 Architectural Variations

We have so far considered a model with a predictive-corrective block at the `fc7` layer that re-initializes every 4 frames. However, different layers of the network capture different information about the video, and evolve at different rates. We investigate these options to gain deeper insight into the model and into the structure of temporal data.

Single block: We begin by experimenting on models with a single predictive-corrective block. We consider placing the block at different layers in the model other than `fc7`, thus asking the model to focus on more low-level (`conv53`) or high-level (`fc8`) visual changes. Table 3 reports the results. We find that placing a predictive corrective block at `fc7` is optimal within the single-block setting. Placing the block at either `conv53` or `fc8` yields a 0.4% and 0.3% respective reduction in mAP. Reasoning about higher-level but non-semantic features proves to be the most effective.

Hierarchical blocks: By placing a single predictive-corrective block, we force the entire model to reinitialize its memory at the same rate. We hypothesize that reinitializing at a faster rate may be important, particularly for predictive-corrective blocks placed at lower levels in the network since the low-level visual features change faster than the more semantic `fc7`. Encouraged by this intuition, we experiment with placing predictive-corrective blocks at multiple layers with different reinitialization rates. We explore a few hierarchical configurations in Table 3. In particular, the "`conv33` every 1, `fc7` every 4" model can be interpreted as predicting and correcting `conv33` activations instead of *pixel* values (as the "`fc7` every 4" model does), which are less sensitive to noise, brightness changes, and camera motion than raw pixels. Indeed, this model outperforms all other configurations, achieving 27.2% mAP.

Effective corrections: We conclude with a qualitative look into the predictions made by our model. In particular, one worry is that the model may be predicting the same action labels across all 4 frames between reinitializations. Fig. 8 shows that this is not the case. The predictive-corrective block is able to successfully notice the changes that occur between frames and update the action predictions. For example, in the first row of Fig. 8 the "jump" action happens 2 frames after reinitialization, and the model successfully corrects its initial prediction.

4.2. Comparison to Prior Approaches

Building off our analysis in Sec. 4.1, we now evaluate our predictive-corrective model on three challenging benchmarks: MultiTHUMOS [55], THUMOS [15], and Charades [37]. Table 3 motivates using the hierarchical "`conv33` every 1, `fc7` every 4" architecture; Table 2 demonstrates that training to reinitialize every 8 frames yields further improvement. Thus we use the "`conv33` every 1, `fc7` every 8" as our predictive-corrective model.

4.2.1 THUMOS and MultiTHUMOS

Setup: THUMOS [15] contains 20 annotated action classes; MultiTHUMOS [55] includes 45 additional action classes annotated on the THUMOS videos. We train models on the training and validation videos for all the MultiTHUMOS actions jointly. We then evaluate on the THUMOS test videos by computing the per-frame mAP over the 20 THUMOS and 65 MultiTHUMOS action classes.

Results: We report results in Table 4. The single-frame model has been shown to be a strong baseline for action recognition, outperforming, e.g., C3D [42] in [37]. On MultiTHUMOS our predictive-corrective model not only outperforms the single-frame baseline by 4.3% mAP (29.7% mAP ours vs 25.4% mAP single-frame), but also compares favorably to the state-of-the art MultiLSTM model [55].

On THUMOS, our model still outperforms the single-frame model by 4.2% (38.9% mAP ours vs 34.7% mAP single-frame), but is not yet on par with MultiLSTM. This may be due to the significantly longer actions in THUMOS, which the LSTM-based model can handle better due to a longer (though less interpretable) memory of the video.

At the cost of efficiency, we can further improve our model by running it in a dense sliding window fashion where the model has a 7 frame history when making a prediction for each frame. With this approach, our model achieves 30.8% mAP on MultiTHUMOS (significantly outperforming MultiLSTM's 29.6% mAP) and 40.9% on THUMOS (only 0.4% behind MultiLSTM at 41.3% mAP).

4.2.2 Charades

Setup: Whereas the THUMOS and MultiTHUMOS datasets contain primarily videos of sports actions, the Charades dataset [37] contains videos of common everyday actions performed by people in their homes. The dataset contains 7,986 untrimmed training videos and 1,864 untrimmed test videos, with a total of 157 action classes. This is a significantly more challenging testbed: first, it contains many more actions than MultiTHUMOS, and second, it is constructed so as to decorrelate actions from scenes.

Method	MultiTHUMOS	THUMOS
Single-frame [55]	25.4	34.7
Two-Stream[3][38]	27.6	36.2
Multi-LSTM [55]	29.6	**41.3**
Predictive-corrective	**29.7**	38.9

Table 4. Comparison of our model with prior art. (Per-frame mAP on MultiTHUMOS and THUMOS test sets.)

Method	Charades
Single-frame	7.9
LSTM (on RGB)	7.7
Two-Stream [35]	**8.9**
Predictive-corrective	**8.9**

Table 5. Comparison of our model with prior work on Charades. Our model matches the accuracy of the two-stream model without using optical flow. (Localization mAP on the Charades test set.)

Results: Our model generalizes to this new domain despite the challenges. We report action localization results (following [36]) in Table 5. Our predictive-corrective model improves from 7.9% mAP of the single-frame baseline and the 7.7% mAP of the LSTM baseline to 8.9% mAP. Further, our model is able to match the accuracy of the two-stream network, without the need for explicitly computing expensive optical flow.[2]

5. Conclusions

We introduced a recurrent predictive-corrective network that maintains an *interpretable* memory that can be dynamically re-initialized. Motivated by Kalman Filters, we exploit redundancies and motion cues within videos to smoothly update our per-frame predictions and intermediate activations within a convolutional network. We perform extensive ablation studies of this model, carefully choosing where to place predictive-corrective blocks, improving accuracy over baselines on the MultiTHUMOS and THUMOS datasets.

Acknowledgements

We would like to thank Rohit Girdhar and James Supančič for reviewing early versions of this paper, and Serena Yeung and Gunnar Sigurdsson for help working with the MultiTHUMOS and Charades datasets. Funding for this research was provided by NSF Grant 1618903, NSF Grant 1208598, and the Intel Science and Technology Center for Visual Cloud Systems (ISTC-VCS).

[2]For completeness, we note that the model does not yet match state-of-the-art results on the Charades benchmark: e.g., [36] achieves 12.5% mAP using global cues and post-processing.

[3]The two-stream number is reported from [55], which uses a single optical flow frame for the flow stream.

References

[1] S. Abu-El-Haija, N. Kothari, J. Lee, P. Natsev, G. Toderici, B. Varadarajan, and S. Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. *CoRR*, abs/1609.08675, 2016. 1

[2] A. Bordes, L. Bottou, and P. Gallinari. Sgd-qn: Careful quasi-newton stochastic gradient descent. *The Journal of Machine Learning Research*, 10:1737–1754, 2009. 3

[3] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010. 3

[4] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376, 2011. 5

[5] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, et al. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231, 2012. 3

[6] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Computer Vision and Pattern Recognition*, 2015. 1

[7] J. T. Enns and A. Lleras. What's next? new evidence for prediction in human vision. *Trends in cognitive sciences*, 12(9):327–333, 2008. 2

[8] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *Computer Vision and Pattern Recognition*, 2016. 1

[9] C. Finn, I. Goodfellow, and S. Levine. Unsupervised learning for physical interaction through video prediction. In *NIPS*, 2016. 2

[10] S. S. Haykin et al. *Kalman filtering and neural networks*. Wiley Online Library, 2001. 4

[11] G. E. Hinton and R. S. Zemel. Autoencoders, minimum description length, and helmholtz free energy. *Neural Information Processing Systems*, 1994. 2

[12] M. Hoai, Z.-Z. Lan, and F. De la Torre. Joint segmentation and classification of human actions in video. In *Computer Vision and Pattern Recognition (CVPR)*, 2011. 2

[13] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In D. Blei and F. Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 448–456. JMLR Workshop and Conference Proceedings, 2015. 3

[14] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013. 2

[15] Y. Jiang, J. Liu, A. R. Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. Thumos challenge: Action recognition with a large number of classes. In *ECCV Workshop*, 2014. 2, 3, 5, 8

[16] S. J. Julier and J. K. Uhlmann. New extension of the kalman filter to nonlinear systems. In *AeroSense'97*, pages 182–193. International Society for Optics and Photonics, 1997. 4

[17] S. Karaman, L. Seidenari, and A. Del Bimbo. Fast saliency based pooling of fisher encoded dense trajectories. In *THUMOS'14 challenge entry*, 2014. 2

[18] A. Karpathy, J. Johnson, and L. Fei-Fei. Visualizing and understanding recurrent networks. In *ICLR Workshop*, 2016. 3

[19] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Computer Vision and Pattern Recognition*, 2014. 2, 6

[20] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber. A clockwork rnn. In *International Conference on Machine Learning*, 2014. 2, 5

[21] L. Ljung. Asymptotic behavior of the extended kalman filter as a parameter estimator for linear systems. *IEEE Transactions on Automatic Control*, 24(1):36–50, 1979. 4

[22] W. Lotter, G. Kreiman, and D. Cox. Deep predictive coding networks for video prediction and unsupervised learning. *CoRR*, abs/1605.08104, 2016. 2

[23] M. Marszalek, I. Laptev, and C. Schmid. Actions in context. In *Computer Vision and Pattern Recognition (CVPR)*, 2009. 2

[24] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016. 2

[25] S. Mereu, J. M. Zacks, C. A. Kurby, and A. Lleras. The role of prediction in perception: Evidence from interrupted visual search. *Journal of experimental psychology: human perception and performance*, 40(4):1372, 2014. 2

[26] A. Oliva and A. Torralba. The role of context in object recognition. *Trends in Cognitive Sciences*, 11(12), 2007. 2

[27] D. Oneata, J. Verbeek, and C. Schmid. The lear submission at thumos 2014. In *THUMOS'14 challenge*, 2014. 2

[28] D. Parikh and C. Zitnick. Human-debugging of machines. *NIPS WCSSWC*, 2:7, 2011. 3

[29] R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318, 2013. 3

[30] B. A. Pearlmutter. Gradient calculations for dynamic recurrent neural networks: A survey. *IEEE Transactions on Neural networks*, 6(5), 1995. 1

[31] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3), 2015. 5

[32] E. Shelhamer, K. Rakelly, J. Hoffman, and T. Darrell. Clockwork convnets for video semantic segmentation. In *ECCV*, 2016. 2, 5

[33] Q. Shi, L. Cheng, L. Wang, and A. Smola. Human action segmentation and recognition using discriminative semi-markov models. *International journal of computer vision*, 93(1):22–32, 2011. 2

[34] Z. Shou, D. Wang, and S.-F. Chang. Temporal action localization in untrimmed videos via multi-stage cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1049–1058, 2016. 5

[35] G. Sigurdsson. Charades dataset. http://allenai.org/plato/charades/, 2017. Accessed: 2017-04-10. 8

[36] G. A. Sigurdsson, S. Divvala, A. Farhadi, and A. Gupta. Asynchronous temporal fields for action recognition. *arXiv preprint arXiv:1612.06371*, 2016. 8

[37] G. A. Sigurdsson, G. Varol, X. Wang, A. Farhadi, I. Laptev, and A. Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *ECCV*, 2016. 2, 8

[38] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Neural Information Processing Systems*, 2014. 1, 2, 8

[39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5

[40] N. Srivastava, E. Mansimov, and R. Salakhutdinov. Unsupervised learning of video representations using LSTMs. In *ICML*, 2015. 1, 2

[41] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. 3

[42] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *International Conference on Computer Vision (ICCV)*, 2015. 1, 2, 8

[43] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *arXiv preprint arXiv:1604.04494*, 2016. 1, 2

[44] S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, and K. Saenko. Sequence to sequence-video to text. In *International Conference on Computer Vision*, 2015. 1

[45] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating visual representations from unlabeled video. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[46] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *Neural Information Processing Systems (NIPS)*, 2016. 2

[47] J. Walker, C. Doersch, A. Gupta, and M. Hebert. An uncertain future: Forecasting from variational autoencoders. In *European Conference on Computer Vision*, 2016. 2

[48] H. Wang and C. Schmid. Action recognition with improved trajectories. In *International Conference on Computer Vision*, 2013. 1, 2

[49] L. Wang, Y. Qiao, and X. Tang. Action recognition and detection by combining motion and appearance features. In *THUMOS'14 challenge entry*, 2014. 2

[50] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015. 1

[51] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: towards good practices for deep action recognition. In *European Conference on Computer Vision*, 2016. 1

[52] S. B. Wang, A. Quattoni, L.-P. Morency, D. Demirdjian, and T. Darrell. Hidden conditional random fields for gesture recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2006. 2

[53] L. Wiskott and T. J. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 14(4):715–770, 2002. 3

[54] T. Xue, J. Wu, K. L. Bouman, and W. T. Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *NIPS*, 2016. 2

[55] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *arXiv preprint arXiv:1507.05738*, 2015. 2, 5, 8

[56] S. Yeung, O. Russakovsky, G. Mori, and L. Fei-Fei. End-to-end learning of action detection from frame glimpses in videos. In *CVPR*, 2016. 1, 2, 5

[57] J. Yuan, Y. Pei, B. Ni, P. Moulin, and A. Kassim. Adsc submission at thumos challenge 2015. In *THUMOS'15 challenge entry*, 2015. 2

[58] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Computer Vision and Pattern Recognition*, 2015. 1, 2, 6

[59] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014. 3, 5