Feedback Loop between High Level Semantics and Low Level Vision

Varun K. Nagaraja Vlad I. Morariu Larry S. Davis
{varun,morariu,lsd}@umiacs.umd.edu

University of Maryland, College Park, MD, USA.

Abstract. High level semantic analysis typically involves constructing a Markov network over detections from low level detectors to encode context and model relationships between them. In complex higher order networks (e.g. Markov Logic Networks), each detection can be part of many factors and the network size grows rapidly as a function of the number of detections. Hence to keep the network size small, a threshold is applied on the confidence measures of the detections to discard the less likely detections. A practical challenge is to decide what thresholds to use to discard noisy detections. A high threshold will lead to a high false dismissal rate. A low threshold can result in many detections including mostly noisy ones which leads to a large network size and increased computational requirements. We propose a feedback based incremental technique to keep the network size small. We initialize the network with detections above a high confidence threshold and then based on the high level semantics in the initial network, we incrementally select the relevant detections from the remaining ones that are below the threshold. We show three different ways of selecting detections which are based on three scoring functions that bound the increase in the optimal value of the objective function of network, with varying degrees of accuracy and computational cost. We perform experiments with an event recognition task in one-on-one basketball videos that uses Markov Logic Networks.

1 Introduction

Computer vision systems are generally designed as feed-forward systems where low level detectors are cascaded with high level semantic analysis. Low level detectors for objects, tracks or short activities usually produce a confidence measure along with the detections. The confidence measures can sometimes be noisy and hence a multitude of false detections are fed in to subsequent analysis stages. To avoid these false detections, it is common practice to discard some detections that are below a particular confidence threshold. Unfortunately, it is difficult to reliably select a threshold a priori given a particular task. The threshold is generally selected to achieve a "reasonable" trade-off between detector precision and recall, since it is generally not possible to find all true detections (high recall) without also hallucinating false alarms (low precision).

High level analysis integrates multiple low level detections together using semantics to discard false detections rather than simply thresholding detector

scores. For example, in an event recognition system for basketball, the low level detections like shot missed and rebound events are related by high level rules of the game which say that a shot missed event is followed by a rebound event. The analysis of high level interactions between detections can improve the confidence in the detections.

High level analysis typically involves constructing a Markov network over the detections, where contextual relationships corresponding to high level knowledge about the image or video are encoded as factors over combinations of detections [1–5]. A detection usually corresponds to one or more nodes in the network and relationships between detections correspond to a factor. In Markov networks of high order, each detection can be part of exponentially many instantiations of a factor and the network size grows rapidly as a function of the number of detections. The problem is further exacerbated by the inference process, whose computational cost is related exponentially to the network complexity. When many detections are hypothesized at low precision, the size of the Markov network becomes unnecessarily high since the inference process sets most of the detections to false.

We tackle the problem of keeping the network size small by incrementally adding only those detections that are most likely to be inferred as true while the rest of them are kept false. We achieve this by adding a feedback loop between the high level and low level stages, where the high level semantics guides the selection of relevant low level detections. There are several advantages to this feedback loop. First, it can locally adjust the thresholds for low level detectors based on the neighboring context. Second, it keeps the network size small and the inference procedure tractable. And third, we can potentially save computation by selectively running the low level procedures like feature extraction and classification only when needed.

The goal of our feedback based incremental technique is to perform inference and obtain the optimal solution of the objective function corresponding to the *full network* (the network obtained when we include all the detections) by unclamping only the relevant detections. We start with detections above a high confidence threshold and clamp the remaining detections to false based on the closed world assumption, the assumption that what is not known to be true is false. We then incrementally select from the remaining detections below the threshold to add to the network. Our proposed feedback loop involves a principled mechanism by which we identify the detections that are most likely to improve the objective function. Motivated by cluster pursuit algorithms [6] for inference, we derive three scoring functions that bound the increase in the objective function with varying degrees of accuracy and computational cost. The first score function yields the exact increase in the objective function, but it requires that the detector has been run everywhere and that inference can be performed exactly; the second bounds the change in the objective function, relaxing the inference requirements; the third provides an even looser bound, but it is least computationally intensive and does not require the low level detector to have processed the candidate detections (which is why we call it the *Blind Score*).

We perform experiments on an event recognition task using one-on-one basketball videos. Morariu and Davis [3] used Markov Logic Networks (MLNs) on this dataset to detect events like Shot Made, Shot Missed, Rebound etc. The inputs are a set of event intervals hypothesized from low level detectors like the tracks of objects. Using the feedback loop technique we show that we can successfully select the most relevant event intervals that were earlier discarded due to thresholding. The experiments show that our score functions can reach the optimal value in fewer iterations with smaller network sizes when compared with using just the low level confidence measures.

2 Related Work

High level context plays an important role in many vision systems like scene segmentation [7], object detection in 2D [4, 8] and 3D [5] and event recognition [1–3]. Usually these systems hypothesize a set of candidate detections using low level detectors and then feed them into the high level model which assigns a label to the candidate detections based on the context. Since low level detectors are not perfect, a multitude of false positives propagate from the low level to the high level. So a high level system is faced with the choice of either dealing with a large model size or having a threshold for the inputs so that model size is contained, but only by discarding true detections that happen to have low confidences.

While many inference techniques work in an incremental fashion to tackle the complexity issues, they do not necessarily behave as a feedback loop and hence do not present with the advantages mentioned earlier. We mention few works here that iteratively add detections while performing inference. In a scene segmentation task, Kumar and Koller [7] hypothesize a set of regions in an image through multiple bottom-up over-segmentations and exploit the high level energy function to iteratively select input regions that are relevant for the task. Zhu et al. [9] use the greedy forward search technique of Desai et al. [10] for inference in their event recognition system. The inference algorithm of Desai et al. first sets the output label for the inputs to the background class. Each input is then scored based on the change in the objective function if it were allowed to be labelled as a non-background class. The top scoring inputs are then iteratively added until convergence. Our feedback loop technique is based on the same idea of greedily reaching the MAP value as quickly as possible but we provide a principled mechanism to performing inference in higher order networks. Also we do not use it just as an incremental technique, but extract more insight from the high level semantics to save computation for the low level module. An interesting characteristic of our feedback technique is that we can potentially run low level processes only when required during the inference.

Apart from the advantages of keeping the inference tractable, a feedback loop can also be useful in other ways. Sun et al. [8] apply a feedback loop for object detection with geometrical context. They jointly infer about the location of an object, the 3D layout of the scene and the geometrical relationships between the

object and the 3D layout. The speciality of their feedback loop is that the object detector module adaptively improves its accuracy in the confidence measures of detections based on the feedback from the scene layout.

The idea of incrementally building a network can be approached in principled ways, including Cutting Plane Inference (CPI) and Cluster Pursuit Algorithms. Many inference problems can be cast as an Integer Linear Program (ILP) which is well suited for CPI. CPI employs an iterative process where the ILP is kept small by adding only the most violated constraints. However, CPI cannot be used for our feedback loop technique where we need to selectively set some detections to false. Sontag et al. [6] propose a cluster pursuit algorithm, an alternative formulation that incrementally adds cliques of variables (called *clusters*) and optimizes the dual function, an objective function obtained through Lagrangian relaxation that is an upper bound on the original (or *primal*) objective function. Their score function for clusters is an approximation to the decrease in the dual value of the objective function after adding a cluster, which is derived from the message passing updates of Globerson and Jaakkola [11]. We use this idea of cluster pursuit algorithm and derive a feedback technique for higher order Markov networks. Our scoring functions use the dual value to calculate approximations for the increase in the primal MAP value after adding a particular cluster.

3 Incremental Inference with Feedback Loop

We consider Markov networks defined over binary nodes $\boldsymbol{x} = \{x_1, \ldots, x_n\}$ with factors $\theta_c(\boldsymbol{x}_c)$ defined over cliques of nodes $\boldsymbol{x}_c$ such that $c_1, \ldots, c_k \subset \{1, \ldots, n\}$. The Maximum A Posteriori (MAP) problem is defined as finding an assignment $\boldsymbol{x}^*$ that maximizes the function

$$\Phi(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{c \in \mathcal{C}} \theta_c(\boldsymbol{x}_c) \tag{1}$$

The nodes x_i are instantiated over candidate detections that are hypothesized by low level detectors. For example, they can be object detections obtained from running single-object detectors. The detector confidence scores output along with the detections are used as unary factors for the nodes. The factors θ_c that involve more than one detection represent the relationships between the detections. For example, they can be spatial relationships like the placement of an object on top of other objects. We obtain a MAP solution by performing inference, that will ultimately label the hypothesized detections as true positives or false positives.

In Markov networks of high order, every newly added detection can become combinatorially linked to other detections through the higher order factors. When many detections are hypothesized at low precision, the size of the Markov network becomes exponentially large and the inference process becomes computationally expensive even though many of the detections are going to be inferred as false.

The goal of our incremental approach for inference is to maximize the function in (1) while keeping the network size small. We achieve this by unclamping

only those detections that are most likely to be labeled as true by the inference. The rest of the detections are clamped to false, and while they always participate in the objective function over the iterations, they are excluded from the network during inference. We first perform inference with an initial network constructed from high confidence detections while the rest are clamped to false. We then calculate scores for the remaining detections based on the initial network. The scores measure the change in the MAP value after adding a detection to the current network. These scores are equivalent to locally adding an offset to the low level detector confidences, based on the feedback, so that the detections appear above the threshold. Another way to interpret this is that the thresholds get locally modified to select the detections that are below the threshold. We then unclamp a selected number of top detections and the process is repeated. When the incremental procedure is stopped, the MAP solution to the current network provides the true/false labels to the active detections and the remaining set of detections are labeled as false.

3.1 Clusters under closed world assumption

We show that incrementally unclamping detections is equivalent to adding clusters of factors. First we partition the Markov network into three clusters as shown in Figure (1). Let f be the set of active detections that are currently in to the network and $\boldsymbol{x_f}$ be the nodes that are instantiated over only the detections from f. The factor θ_f is defined over just the nodes $\boldsymbol{x_f}$. Let g be the set of one or more detections that is to be unclamped in a given iteration and $\boldsymbol{x_g}$ be the nodes instantiated over at least one detection from g and any other detections from f. The factor θ_g is defined over nodes $\boldsymbol{x_g}$ and other nodes from $\boldsymbol{x_f}$ that it shares with θ_f. Let h be the remaining set of detections and $\boldsymbol{x_h}$ be the nodes that are grounded over at least one detection from h and any other detections from $f \cup g$. The factor θ_h is defined over $\boldsymbol{x_h}$ and the other shared nodes with θ_f and θ_g. The overall objective function expressed as a sum of these clusters is

$$\begin{aligned} \Phi(\boldsymbol{x}) = \theta_f(x_{f1}, x_{f2}, x_{f3}, x_{f4}) + \theta_g(x_{g1}, x_{g2}, x_{f2}, x_{f3}) \\ + \theta_h(x_{h1}, x_{g2}, x_{f3}, x_{f4}) \end{aligned} \tag{2}$$

Under the closed world assumption, any detection that is not included in the Markov network due to thresholding is assumed to be false. To satisfy this condition during the incremental process, we need to repartition the objective function (2). During every iteration of the process, we have a Markov network that includes a set f of active detections. The remaining detections from g and h are not yet added and hence the nodes instantiated over these detections must be clamped to false. The associated factors are projected on to the current network after setting the nodes of the excluded detections to false. The resulting objective function is

$$\begin{aligned} \Phi_{\text{cur}}(\boldsymbol{x}_{\text{cur}}) = \theta_f(x_{f1}, x_{f2}, x_{f3}, x_{f4}) + \theta_g(x_{g1} = 0, x_{g2} = 0, x_{f2}, x_{f3}) \\ + \theta_h(x_{h1} = 0, x_{g2} = 0, x_{f3}, x_{f4}) \end{aligned} \tag{3}$$

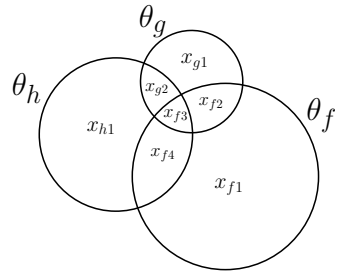


Fig. 1: The shared nodes between clusters in a partitioning of a Markov network. The set f contains active detections that are currently in the network and $\boldsymbol{x_f}$ are the nodes that are instantiated over only the detections from f. The set of factors $\theta_f(\boldsymbol{x_f})$ is defined over the nodes $\boldsymbol{x_f}$. Similarly, g is the set of detections to be unclamped at an iteration and h is the set of detections that are still clamped to false.

To calculate a score for the set of detections in g, we need the objective function to include these detections in the active set while all other remaining detections from h are still clamped to false. This gives rise to the objective function

$$\begin{aligned}\Phi'(\boldsymbol{x}) = \theta_f(x_{f1}, x_{f2}, x_{f3}, x_{f4}) + \theta_g(x_{g1}, x_{g2}, x_{f2}, x_{f3}) \\ + \theta_h(x_{h1} = 0, x_{g2}, x_{f3}, x_{f4})\end{aligned} \tag{4}$$

Hence, the cluster of factors that need to be added to the current network during an iteration is given by

$$\Phi_{\text{new}}(\boldsymbol{x}_{\text{new}}) = \Phi' - \Phi_{\text{cur}}(x_{\text{cur}}) \tag{5}$$

$$\begin{aligned}= \theta_g(x_{g1}, x_{g2}, x_{f2}, x_{f3}) - \theta_g(x_{g1} = 0, x_{g2} = 0, x_{f2}, x_{f3}) \\ - \theta_h(x_{h1} = 0, x_{g2} = 0, x_{f3}, x_{f4}) + \theta_h(x_{h1} = 0, x_{g2}, x_{f3}, x_{f4})\end{aligned} \tag{6}$$

We now propose three score functions that measure the change in the MAP value after adding the cluster $\Phi_{\text{new}}(x_{\text{new}})$ to $\Phi_{\text{cur}}(x_{\text{cur}})$, with varying degrees of accuracy and computational cost.

3.2 Detection scoring function

We define the score for a detection based on the change in the MAP value after adding the detection to the current network. If we are adding the detection in g, the score is given by

$$score(g)_{exact} = \Delta\Phi = \max\left[\Phi_{\text{cur}}(x_{\text{cur}}) + \Phi_{\text{new}}(x_{\text{new}})\right] - \max\left[\Phi_{\text{cur}}(x_{\text{cur}})\right] \tag{7}$$

We also propose an upper bound to the exact score - $score(g)_{upper}$, that is derived based on the ideas of cluster pursuit algorithm of Sontag et al. [6]. We first obtain a dual of the MAP problem through Lagrangian relaxation.

The MAP problem is now equivalent to minimizing the dual objective function since the dual value is an upper bound on the primal MAP value. We then use the message passing algorithm of Globerson et al. [11] to obtain the message update equations for the dual variables. Similar to Sontag et al. [6], we obtain an approximation to the new dual value after adding a cluster to the current network, by performing one iteration of message passing. Since the dual value is an upper bound on the primal MAP value, the new decreased dual value gives an upper bound for the exact score.

Proposition 1 (Upper Bound Score). *An upper bound on the change in the MAP value (7) after adding a cluster is given by*

$$\Delta\Phi \leq score(g)_{upper} \tag{8}$$

$$= \frac{1}{|s|} \sum_{i \in s} \max_{x_i} \left(\max_{x_{cur \setminus i}} \Phi_{cur}(x_{cur}) + \max_{x_{new \setminus i}} \Phi_{new}(x_{new}) \right) - \max \Phi_{cur}(x_{cur}) \tag{9}$$

where s is the set of nodes in the intersection of the sets x_{cur} and x_{new}.

The proof can be found in the supplementary material. The first term in the upper bound score is equivalent to averaging the MAP values obtained by enforcing same assignment for one shared node at a time. The upper bound score can be calculated efficiently using an inference algorithm that calculates max-marginals with only a little computation overhead (eg. dynamic graph cuts [12]) and hence can avoid performing repeated inference to calculate the exact score.

We derive another approximation to the score function called the *Blind Score* since it is dependent only on the max-marginals of the current network and does not involve the max-marginals of the new cluster to be added. It is obtained as a lower bound to the upper bound score (not the exact score).

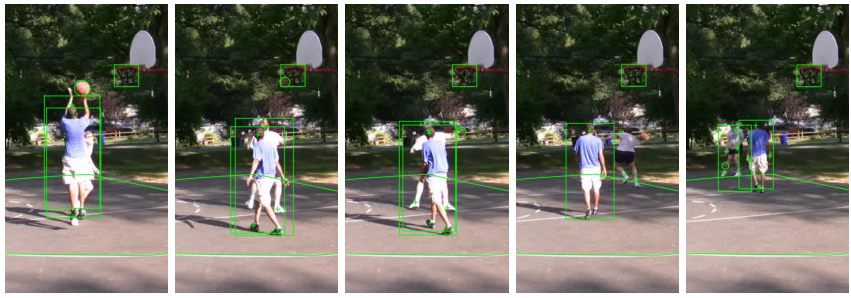
Proposition 2 (Blind Score). *A lower bound to the upper bound score (9) is given by*

$$score(g)_{upper} \geq score(g)_{blind} \tag{10}$$

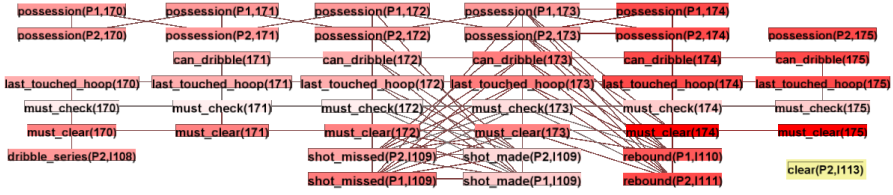
$$= \frac{-1}{|s|} \sum_{i \in s} \left| \max_{x_i = 0, x_{cur \setminus i}} \Phi_{cur}(x_{cur}) - \max_{x_i = 1, x_{cur \setminus i}} \Phi_{cur}(x_{cur}) \right| \tag{11}$$

where s is the set of nodes in the intersection of the sets x_{cur} and x_{new}.

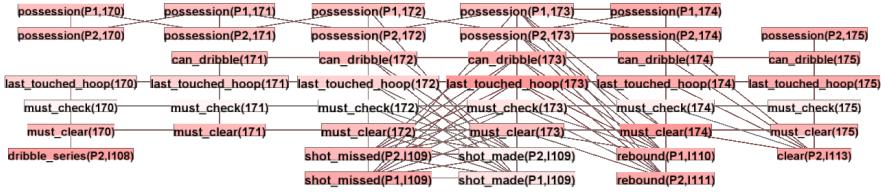
The proof can be found in the supplementary material. This score measures the average of the difference in max-marginals of the shared nodes. It indicates the susceptibility of the shared nodes in the current network to change their values when a new cluster is added. The score is low if the absolute difference in the max-marginals of the shared variables is high. This indicates that the current network has low uncertainty (or strong belief) in the assigned values to the shared variables. Similarly the score is high if the absolute difference in the max-marginals is low. This indicates that the network has high uncertainty in the assignments to the shared variables and that is where we need more evidence/observations.



(a) A sequence of events which shows a shot being missed by Player1 and the rebound received by Player2. When Player2 is clearing the ball, the track goes missing for a while and hence the confidence measure for that clear event is low.



(b) Applying an initial threshold for Clear events does not include the highlighted Clear event. However the corresponding Shot Missed event by Player1 is included in the network. The absolute difference in the max-marginals represents *certainty* of a node assignment and hence the negative of that difference represents *uncertainty*. Here, darker colors indicate high uncertainty. When the Clear event is missing, the network is highly uncertain right after the Shot Missed event.



(c) The node assignments become more certain after adding the missing Clear event.

Fig. 2: Visualization of the Feedback Loop

Since the blind score is independent of max-marginals of the new cluster, it does not need the confidence score of a detection which is usually used as a unary potential in the new cluster. This can save computation for the low level detectors by avoiding expensive procedures like feature extraction and classification throughout an image/video and instead run them only when it is needed by the inference. However, the blind score needs to know the shared variables (s) between the new cluster and the current network. This corresponds to determining the locations where the detector would be run and these are usually easy to obtain for sliding-window approaches. For example, to perform 3D object detection, Lin et al. [5] first generate candidate cuboids without object class information which fixes the structure of their network and hence tells us the shared

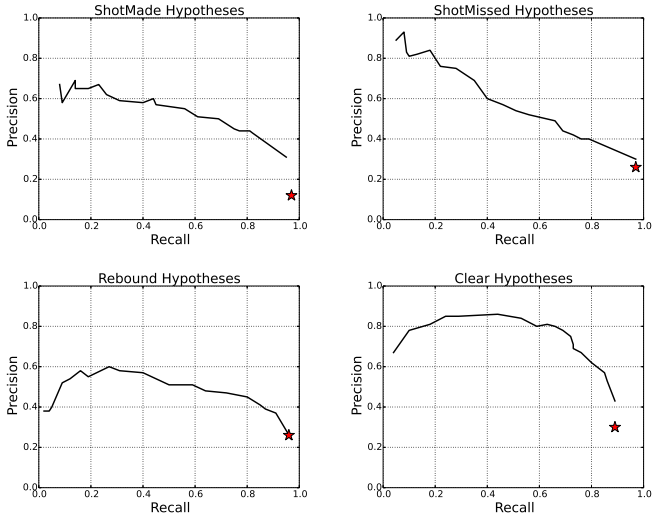


Fig. 3: PR curves for the newly hypothesized events with continuous confidence measures. The red star shows the operating point of Morariu et al. [3] in their feed-forward approach.

variables for any cluster. They then extract features for generating unary potentials and use it in a contextual model to assign class labels to the hypothesized cuboids. If we use the blind score during the inference, we can potentially save computation by not extracting features for cuboids that are likely to be labeled as false. Figure (2) illustrates our feedback loop technique using an example from the basketball dataset of Morariu et al. [3].

4 Experiments

4.1 One-on-One basketball dataset

The one-on-one basketball dataset used by Morariu et al. [3] contains tracks of players and ball along with court annotations for seven basketball videos. The are eight events of interest: Check, Out Of Bounds, Shot Made, Shot Missed, Rebound, Clear, Dribble and Steal. They use a Markov Logic Network (MLN) [13] to represent high level rules of the game which interrelates the various events. The inputs to the MLN are candidate events hypothesized by low level detectors which use the tracks of players and the ball.

4.2 Hypothesizing candidate events

In the MLN used by Morariu et al. [3], each event was hypothesized with just two discrete confidence values. However, continuous confidence measures are required for the events to better tie them to reality. We hypothesize a new

	Morariu et al. [3]			Ours		
	P	R	F1	P	R	F1
Check	0.84	0.89	0.87	**0.86**	**0.90**	**0.90**
Clear	**0.86**	0.61	0.71	0.81	**0.82**	**0.82**
Dribble	**0.81**	0.75	0.78	0.79	**0.82**	**0.80**
OutOfBounds	**0.88**	**0.66**	**0.75**	0.80	0.62	0.70
Rebound	0.62	0.72	0.67	**0.82**	**0.84**	**0.83**
ShotMade	0.64	0.86	0.73	**0.87**	**0.87**	**0.87**
ShotMissed	0.67	0.79	0.72	**0.81**	**0.85**	**0.83**
Steal	0.08	**0.50**	**0.13**	**0.25**	0.25	0.12
Overall	0.72	0.75	0.74	**0.81**	**0.83**	**0.82**

Table 1: Comparison of MLN Recognition Performance using all the hypothesized intervals without thresholding. We can see that the continuous confidence measures for input events play a significant role in improving the performance.

set of candidates with continuous confidence measures for the Shot Made, Shot Missed, Rebound and Clear events and copied the other events (Check, Dribble, Out Of Bounds, Steal) from their dataset. The confidences are obtained based on observations like ball near a player, ball seen inside the hoop, player being inside the two point area, etc. The PR curves of the event hypotheses is shown in Figure (3). Since our modified observation model introduces higher uncertainty in event interval endpoints, we also make few minor modifications to the original MLN to make it robust to the overlapping endpoints of different event intervals.

We first test the importance of continuous confidences in the feed-forward setting by feeding in all the hypothesized intervals to the MLN without thresholding. The confidence measures are used as unary potentials for event predicates in the MLN. Inference is then performed to obtain a MAP assignment for the ground MLN, which labels the candidate events as true or false based on the high level context of the game. The results are shown in Table (1). We see that the confidence measures play a significant role in improving the event recognition performance.

We have implemented our system as an extension of Alchemy [14], a software package for probabilistic logic inference. The MAP problem for MLNs is framed as an Integer Linear Program (ILP) [15] and we integrated our system with the Gurobi ILP solver [16] for performing inference.

4.3 Incrementally adding events with feedback loop

We demonstrate the feedback loop technique by incrementally adding one type of event, the Clear event. The confidence values for the Clear event are scaled between 0.5 and 1. We initialize the network with all the event intervals except for Clear which is thresholded at 0.75. We then run four iterations of the feedback loop and in each iteration, we add a certain number of top ranking Clear events from the remaining set. There are five different kinds of scores that we experiment with: $score(g)_{exact}$, $score(g)_{upper}$, $score(g)_{blind}$, observation score and random score. The observation and random scores are baseline approaches

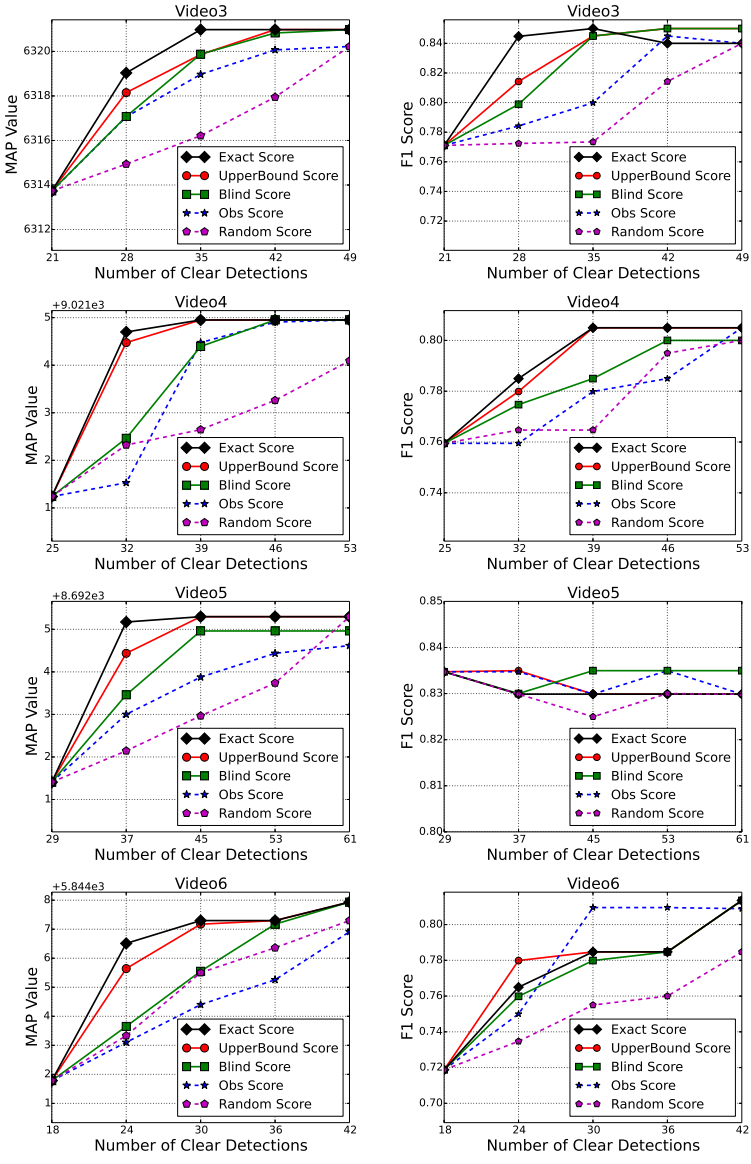


Fig. 4: Feedback based scores achieve better solutions with fewer detections; We apply an initial threshold on the Clear events and incrementally add the remaining events using the feedback based scores. We measure the exact MAP value of the Markov network along with the $f1$ score corresponding to the ground truth. The plots start at the same initial value for all the five scoring methods since the initial network contains the same set of events. Our feedback based scores achieve better solutions with fewer detections than the baselines - observation score and random score.

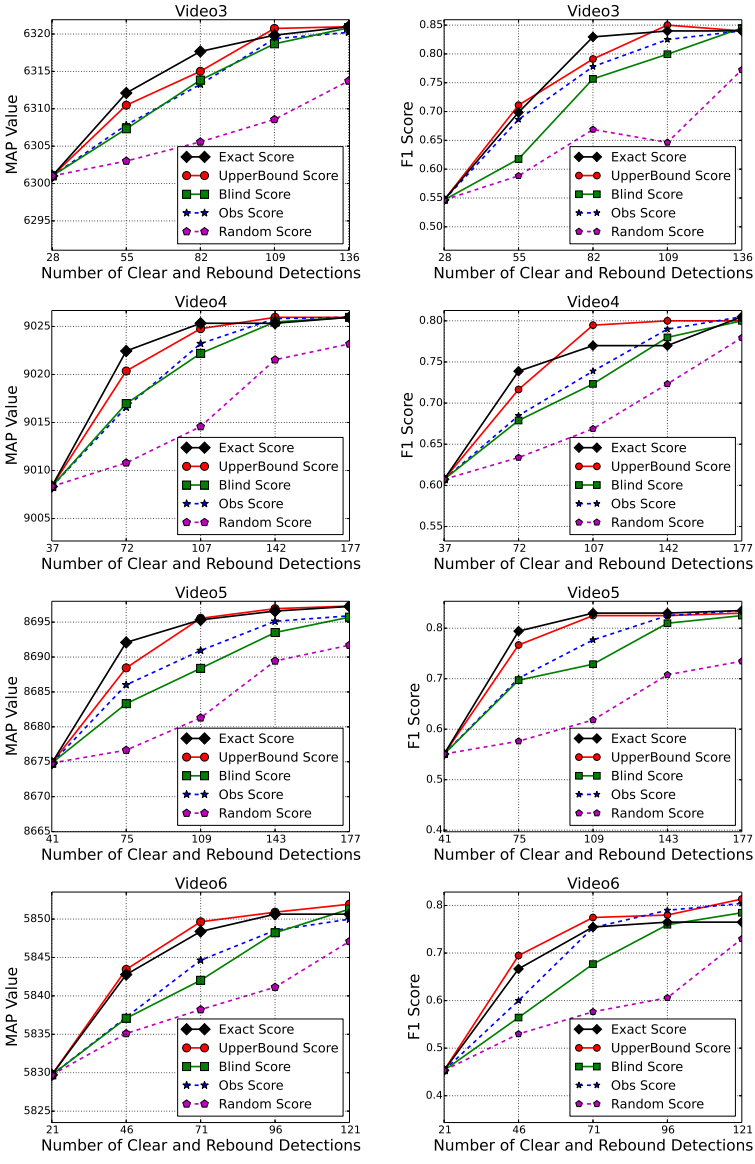


Fig. 5: We apply threshold on both the Rebound and Clear events for initial network and then incrementally add both events at every iteration. We still see that the exact score and the upper bound score reach better solutions with fewer detections than the observation score. However, the blind score falls slightly below the observation score since it depends only on the current network and the context in the current network is weak due to fewer events.

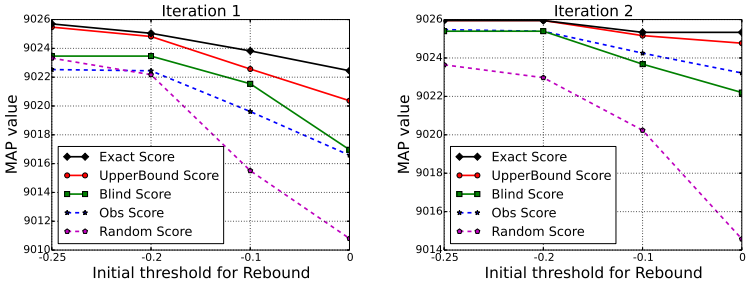
to incrementally adding constants without using a feedback loop. The observation score is the confidence measure that comes from the low level detectors. By adding constants based on their observation score, we are effectively reducing the threshold uniformly throughout the video. The random score is basically selecting a certain number of Clear events randomly and adding them without looking at either the confidence measures or the context in the main network.

The results are shown in Figure (4). Among the seven videos from the dataset, four of the them are large enough to add intervals in an iterative manner. We show the plots of MAP value and also the $f1$ scores against the number of Clear detections in the current network. The plots start at the same initial value for all the five scoring methods since the initial network contains the same set of detections. The goal of our feedback technique is to reach the final MAP value in few iterations by adding only the relevant detections while keeping the rest of them false. The MAP values increase faster with all of our three feedback based score functions when compared to the observation score. The exact score is the quickest followed by the upper bound score and then the blind score. The plots of $f1$ scores also show that we can reach the best possible value with fewer detections using feedback based score functions implying that they select the most relevant events from the missing ones. We observe that the blind score performs well when compared with the observation score. This indicates that the context in the main network has a huge impact on what needs to be added to improve the MAP value.

We also experiment with jointly thresholding the Rebound event along with the Clear event. The Rebound events are scaled between -0.25 to 0.1 and we choose a threshold of 0 for the initial network. The Clear events are scaled between 0.5 to 1 and we choose a threshold of 0.75. We then proceed to iteratively add the remaining Rebound and Clear events. The results in Figure (5) show that the exact score and upper bound score can reach the best possible MAP value and $f1$ score by adding fewer detections. However the plot for blind score falls below that of the observation score. By increasing the threshold on the Rebound event, the strength of context in the main network is weakened and hence the blind score which is dependent on just the current network starts to perform poorly.

4.4 Effect of initial threshold

To observe the effect of initial threshold, we experimented with four different initial thresholds for the Rebound event. Like before, the Rebound events are scaled between -0.25 to 0.1 and the Clear events are scaled between 0.5 to 1. We choose a threshold of 0.75 for Clear events and vary the initial threshold for Rebound events starting from the lowest, which is -0.25 (includes all the Rebound events) and increase up to the value 0 which is high enough to weaken the context. As the initial threshold is increased for the Rebound events, the initial network becomes sparse weakening the context in the initial network. Figure (6a) shows that a higher threshold decreases the MAP value achieved in the first iteration of adding events to initial network. The blind score is affected



(a) First iteration of adding Rebound and Clear events

(b) Second iteration of adding Rebound and Clear events

Fig. 6: Effect of initial threshold for the Rebound event in video 4; The confidence scores for the Clear events are scaled between 0.5 to 1 and the Rebound events between -0.25 to 0.1. We fix the initial threshold for Clear event at 0.75 and vary the threshold for Rebound from -0.25 to 0. We observe that a higher threshold for Rebound event in the initial network decreases the MAP value that is achieved in the first iteration of adding Rebound and Clear events to the initial network. The blind score continues to perform poorly in later iterations at higher initial threshold due to weak context in the initial network. However, the exact score and the upper bound score are still stable with respect to the initial threshold.

the most since it is dependent only on the current network. It continues to perform poorly in later iterations (Figure (6b)) at higher initial threshold for the Rebound event. Hence, it is important to select a reasonably high threshold that allows enough number of events in the initial network without increasing the network size.

5 Conclusion

We propose a computational framework for a feedback loop between high level semantics and low level detectors in a computer vision system, where we use the information in the high level model to select relevant detections from a set of candidate hypotheses. We start with high confidence detections and then iteratively add only those detections to the model that are most likely to be labeled as true by the high level model. This helps us keep the model size small especially in the presence of many noisy detections. We develop the framework for higher order Markov networks and propose three feedback based scoring functions to rank the detections. We show through our experiments on an event recognition system that the feedback loop can construct smaller networks with fewer detections and still achieve the best possible performance.

Acknowledgement: This research was supported by contract N00014-13-C-0164 from the Office of Naval Research through a subcontract from United Technologies Research Center.

References

1. Tran, S., Davis, L.: Event Modeling and Recognition using Markov Logic Networks. In: ECCV. (2008)
2. Brendel, W., Fern, A., Todorovic, S.: Probabilistic Event Logic for Interval-based Event Recognition. In: CVPR. (2011)
3. Morariu, V., Davis, L.: Multi-agent Event Recognition in Structured Scenarios. In: CVPR. (2011)
4. Choi, M., Torralba, A., Willsky, A.: A Tree-based Context Model for Object Recognition. PAMI **34** (2012) 240–52
5. Lin, D., Fidler, S., Urtasun, R.: Holistic Scene Understanding for 3D Object Detection with RGBD Cameras. In: ICCV. (2013)
6. Sontag, D., Meltzer, T., Globerson, A.: Tightening LP Relaxations for MAP using Message Passing. In: UAI. (2008)
7. Kumar, M.P., Koller, D.: Efficiently Selecting Regions for Scene Understanding. In: CVPR. (2010)
8. Sun, M., Bao, S.Y., Savarese, S.: Object Detection using Geometrical Context Feedback. IJCV (August 2012)
9. Zhu, Y., Nayak, N., Chowdhury, A.R.: Context-Aware Activity Recognition and Anomaly Detection in Video. In: CVPR. (2013)
10. Desai, C., Ramanan, D., Fowlkes, C.: Discriminative Models for Multi-Class Object Layout. In: ICCV. (2009)
11. Globerson, A., Jaakkola, T.: Fixing Max-Product: Convergent Message Passing Algorithms for MAP LP-Relaxations. In: NIPS. (2007)
12. Kohli, P., Torr, P.: Measuring Uncertainty in Graph Cut Solutions - Efficiently Computing Min-Marginal Energies Using Dynamic Graph Cuts. In: ECCV. (2006)
13. Richardson, M., Domingos, P.: Markov Logic Networks. Machine Learning (January 2006)
14. Kok, S., Sumner, M., Richardson, M., Singla, P.: The Alchemy System for Statistical Relational (2009)
15. Noessner, J., Niepert, M., Stuckenschmidt, H.: RockIt: Exploiting Parallelism and Symmetry for MAP Inference in Statistical Relational Models. In: AAAI. (2013)
16. Gurobi-Optimization-Inc.: Gurobi Optimizer Reference Manual (2013)