Combining the Right Features for Complex Event Recognition

Kevin Tang Bangpeng Yao Li Fei-Fei Daphne Koller
Computer Science Department, Stanford University
{kdtang,bangpeng,feifeili,koller}@cs.stanford.edu

Abstract

In this paper, we tackle the problem of combining features extracted from video for complex event recognition. Feature combination is an especially relevant task in video data, as there are many features we can extract, ranging from image features computed from individual frames to video features that take temporal information into account. To combine features effectively, we propose a method that is able to be selective of different subsets of features, as some features or feature combinations may be uninformative for certain classes. We introduce a hierarchical method for combining features based on the AND/OR graph structure, where nodes in the graph represent combinations of different sets of features. Our method automatically learns the structure of the AND/OR graph using score-based structure learning, and we introduce an inference procedure that is able to efficiently compute structure scores. We present promising results and analysis on the difficult and large-scale 2011 TRECVID Multimedia Event Detection dataset [17].

1. Introduction

As recent research in video understanding has shifted to classifying complex events like "Attempting a board trick" [17], it is now very difficult for a single feature to capture the information required to discriminate between different complex event categories. Although better feature descriptors have been developed to help characterize videos, it is commonly observed that combining a set of diverse and complementary features is the best approach to this problem [14]. By extracting features from frames of video, we have a whole suite of features available from the image research community that we can leverage. In addition, there are also video features that utilize the temporal aspect of videos. Given these features, the problem we seek to address is finding the optimal way to combine them together for effective complex event recognition.

When combining features, there are two important intuitions we would like to capture.

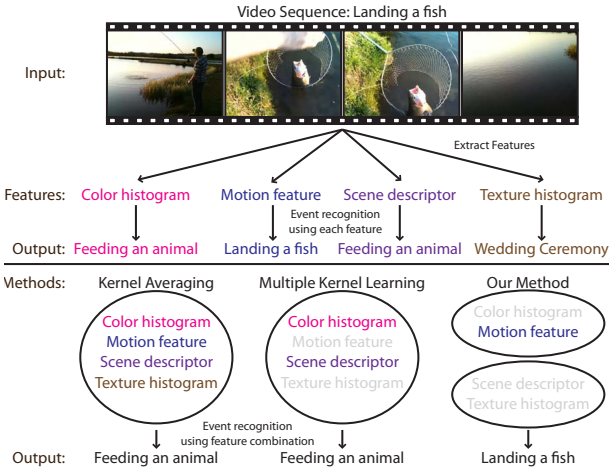

Figure 1. Compared to standard methods for feature combination, our method performs feature selection (grayed features) and considers sets of features independently (separate bubbles). This allows us to select the right features to use and correctly recognize the video sequence as "Landing a fish".

- **Be selective of features to combine.** If we are trying to recognize events of the class "Landing a fish", the motion features would be very important, as we'd like to find the quick reeling action associated with fishing. On the other hand, if we are trying to recognize events of the class "Wedding ceremony", motion features may not be as useful, as there may not be distinctive motion cues that indicate a wedding. Instead, scene descriptors may be better in this case, perhaps to help indicate whether or not we are in a church.

- **Consider sets of features independently.** Different sets of features may be able to provide complementary information when considered separately, as opposed to considering all features together. For example, in "Landing a fish" videos, the information from color and scene may not be complementary, as the presence of blue pixels may repeat information already conveyed by the scene. However, independently combined with other types of features such as motion and

texture, the two features could provide complementary information that could be used together.

Together, these intuitions can help alleviate the limitations in a conventional feature combination approach where all of the features are combined or considered simultaneously. As shown in Figure 1, standard methods like kernel averaging [5] do not perform feature selection, and methods like Multiple Kernel Learning (MKL) [6] consider all features together in a single combination, making it difficult to discover complementary sets of features.

To capture these intuitions, we introduce a novel method for feature combination that represents feature combinations using an AND/OR graph structure, with nodes in the graph representing combinations of different sets of features. The presence of OR nodes allow us to to be selective of the features we want to combine for each class, and the hierarchical structure of the AND/OR graph structure allows us to consider sets of features independently to better discover complementary information. Our method is able to constrain and search the large space of possible AND/OR graph structures for the optimal structure, and we introduce an approximate inference procedure that is able to efficiently compute structure scores. We present convincing results on the 2011 TRECVID Multimedia Event Detection dataset [17] that illustrate the benefits of our approach.

2. Related Work

Many recent works in video understanding have focused on complex event recognition in large-scale datasets [17], which is the focus of this paper. Several works have observed significant performance gains from combining multiple feature types [9, 14, 18, 23], whether through early or late fusion. However, these works consider standard feature combination techniques as a means to improve performance. In our method, we would like to not only improve performance, but also understand and incorporate our intuitions on how features should be selected.

The standard approach to combining features is Multiple Kernel Learning (MKL), which has been used for various tasks in computer vision including object categorization [20], object detection [21], multi-class object classification [5], and complex event recognition [14]. A good overview of MKL with several representative methods is given in [6]. Of the works within MKL, most similar to our method are works that consider a hierarchical combination of kernels [1, 8]. In [1], the author considers hierarchical multiple kernel learning using kernels that can be decomposed into a large sum of separate basis kernels. Our work is different in that we make no restrictions on our kernels, and consider combining a smaller set of well-engineered features. The work of [8] considers semantic kernel forests constructed with human knowledge, and introduces a novel regularizer that exploits the hierarchical structure of their semantic kernels. In contrast, we automatically learn the structure of our hierarchy without human supervision.

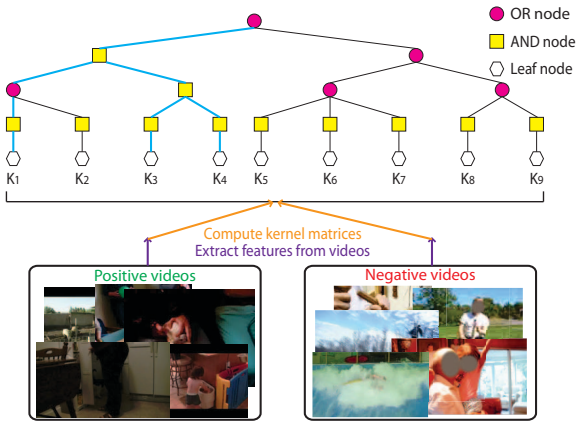

Figure 2. Example of an AND/OR graph structure. The LEAF nodes encode the input kernel matrices, which are then combined using AND/OR nodes up to the root node. The blue edges indicate a possible configuration for the graph.

Our method utilizes the AND/OR graph structure as a representation for combining features. The AND/OR graph structure has been used for many different applications in computer vision [2, 3, 7, 24, 25]. In [2], the authors use an AND/OR graph to infer composite cloth templates. The works of [3, 25] use a pre-specified AND/OR graph to perform object detection, segmentation, and parsing. In [7], the AND/OR graph is used as a storyline model that encodes storyline variation in videos. The authors of [24] use an AND/OR structure to represent grouplets, discriminative features that encode structured image information.

3. Model Representation

Given a training set of N videos and their corresponding binary class labels $\boldsymbol{y} \in \{-1, 1\}$, we can compute a set of m features for each of our videos. Using a kernel function $k_i(x, x')$ that defines a measure of similarity between a pair of instances using feature type i, we can compute the kernel function for all pairs of training instances to obtain a training kernel matrix for each feature: $\boldsymbol{K} = \{K_1, K_2, \ldots, K_m\}$. Kernel matrices can be used to train classifiers like the Support Vector Machine (SVM), and allow us to efficiently represent high dimensional feature spaces. During testing, we can compute the kernel function between our testing instances and training instances, and classify examples accordingly.

Our goal is to devise a method to find a combination of these kernel matrices that can perform effective recognition for a particular event class. Because we associate features with kernel matrices, the problem of kernel com-

bination translates naturally to feature combination. We introduce a method that is selective of these kernel matrices, and simultaneously considers different sets of them independently. Our method uses an AND/OR graph structure to represent the possible combinations, which we describe in detail below.

3.1. AND/OR model

The AND/OR graph structure is represented by a graph $G = (V, E)$, where V and E denote the set of vertices and edges. There are three types of nodes in V, denoted as "AND", "OR", and "LEAF" nodes. The edge set E consists of vertical edges that define the topological structure of the graph, connecting nodes between adjacent layers. We define V_{AND}, V_{OR}, and V_{LEAF} to be the sets of AND, OR, and LEAF nodes in our graph, respectively. We define T_{V_i} to be the child nodes of node $V_i \in V$. Since we do not consider horizontal edges in E, the structure of our AND/OR graph is a tree. We define $V_{root} \in V$ to be the root node of the tree. An example of an AND/OR graph structure is shown in Figure 2.

Each node in the AND/OR graph is a variable that encodes a kernel matrix. The LEAF nodes encode the base kernel matrices $\{K_1, K_2, \ldots, K_m\}$ from our original features at the lowest layer of the graph, and the root node encodes the final kernel matrix to be used for recognition at the highest layer of the graph. Because each LEAF node is just equal to a kernel matrix for one of our original features, the number of LEAF nodes is equal to m.

In our model, there are three types of potentials that define the energy of a particular assignment of kernel matrices $\boldsymbol{v} = \{v_1, v_2, \ldots, v_{|V|}\}$ to our nodes. The first potential captures the behavior of an AND node in the graph, forcing the node to average the kernels of its children. With $V_i \in V_{AND}$:

$$\psi^{AND}(V_i = v_i) = \infty \cdot \mathbf{1}\left[v_i \neq \frac{1}{|T_{V_i}|} \sum_{u \in T_{V_i}} u\right] \quad (1)$$

The second potential captures the behavior of an OR node in the graph, forcing it to select a single kernel from its children. With $V_i \in V_{OR}$:

$$\psi^{OR}(V_i = v_i) = \infty \cdot \mathbf{1}\left[v_i \notin T_{V_i}\right] \quad (2)$$

Finally, the third potential captures the strength of the root node V_{root} in the graph:

$$\psi^{ROOT}(V_{root} = v_{root}) = S(v_{root}) \quad (3)$$

where $S(u)$ is a scoring function that uses the kernel defined at node u to compute the cross-validated average precision on the training data using an SVM. The root node V_{root} also appears in ψ^{AND} or ψ^{OR}, depending on its node type.

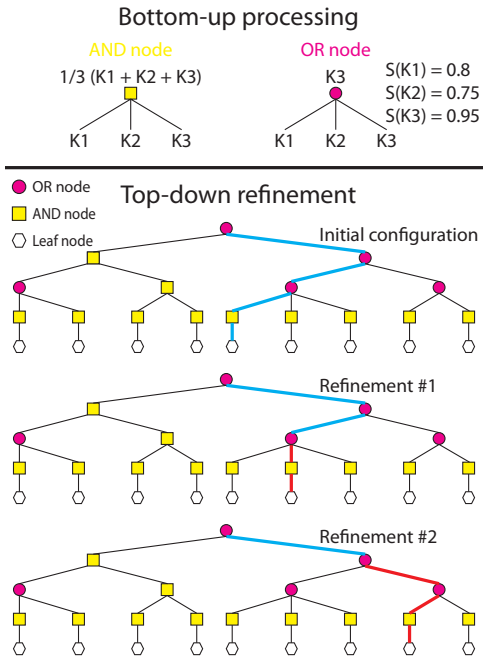

Figure 3. Illustration of the inference procedure. In the bottom-up processing stage, we construct an initial configuration by assigning kernel matrices to each node based only on their children nodes. In the top-down refinement stage, we consider global moves to the initial configuration that had comparable scores in the bottom-up processing stage.

Combining the potentials, we can define the energy of a particular assignment $\boldsymbol{v}$ of kernel matrices to nodes as:

$$\begin{aligned} E(\boldsymbol{v}) = &\sum_{V_i \in V_{AND}} \psi^{AND}(V_i = v_i) \\ &+ \sum_{V_i \in V_{OR}} \psi^{OR}(V_i = v_i) \\ &- \psi^{ROOT}(V_{root} = v_{root}) \end{aligned} \quad (4)$$

Intuitively, if V_i is an AND node in the graph, then it averages the kernels of its children T_{V_i}. If V_i is an OR node in the graph, then it selects a kernel amongst its children T_{V_i}. These restrictions give us a set of possible configurations of the graph, where a configuration is defined as an assignment $\boldsymbol{v}$ that results in a non-infinite energy. Since the behavior of the AND nodes is deterministic, the number of possible configurations is only dependent on the number of OR nodes in the graph.

Note that the space of possible assignments $\boldsymbol{v}$ is not actually the space of all kernel matrices, as nodes in the graph are restricted to combinations of the kernel matrices in the LEAF nodes. Thus, a configuration can be seen as a parse of the graph (blue edges in Figure 2), where we can trace the kernels combined for each node down to the LEAF nodes. Using this interpretation, we can define the magnitude of a

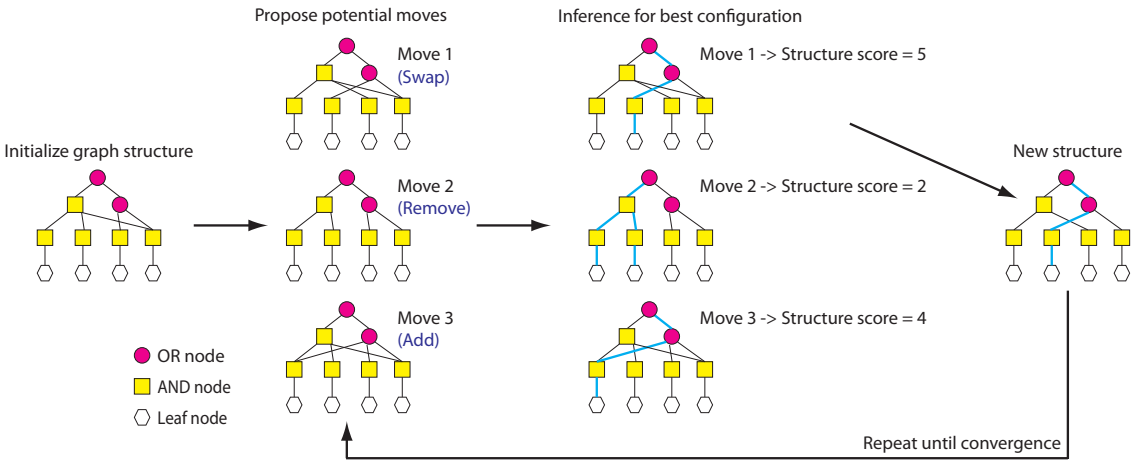

Figure 4. Illustration of the entire learning pipeline. After initializing an AND/OR graph structure, we propose a set of potential moves in the space of possible structures. Then, we perform inference for each of these structures, resulting in the best configuration, which is then used to compute the structure score for each move. Finally, we select the move with the highest score as the new structure. This process is then repeated until convergence, or a maximum number of iterations is reached.

node $|V_i|$ as the number of LEAF nodes that are combined into the kernel for node V_i.

4. Inference

The inference problem seeks to find the assignment of kernel matrices $\boldsymbol{v} = \{v_1, v_2, \ldots, v_{|V|}\}$ that minimizes the energy function $E(\boldsymbol{v})$ in Equation 4. Since the behavior of the AND nodes is deterministic, our goal in inference is to choose the children nodes that the OR nodes select. However, this is difficult because ψ^{ROOT} computes a score based on the kernel at the root node, which couples the decisions of all nodes. Thus, the decisions for the OR nodes cannot be made locally as they could affect the kernel combination at the root node in different ways.

In order to perform efficient inference, we propose an approach inspired by [2, 3] that combines a bottom-up processing stage that proposes configurations for subtrees with a top-down refinement stage that considers a global set of moves over the entire graph.

Bottom-up processing. We start from the nodes in the lowest layer and work our way up to the root node. For each OR node $V_i \in V_{OR}$, we assume that the best kernel assignment v_i is the child $u \in T_{V_i}$ that achieves the best score:

$$v_i = \arg\max_{u \in T_{V_i}} S(u) \tag{5}$$

With this approximation, we can compute the kernel assignments for the OR nodes locally from their children. Combined with the AND nodes, which can also be computed from their children, we have effectively decomposed the inference operation so that local estimates can be made, as seen in the top section of Figure 3. Using this approximation, we can build our configuration from the bottom-up to obtain a kernel assignment for the entire AND/OR graph.

Top-down refinement. In this stage, we would like to refine our approximate configuration from the bottom-up stage. Given the current configuration, we consider global changes to the configuration that may be able to decrease the energy function, as shown in the bottom section of Figure 3. To limit the space of possible refinements, we only consider changing children of OR nodes for which the local estimates from Equation 5 were close in score.

5. Structure Learning

Our goal in structure learning is to find the best AND/OR graph structure and configuration for a particular class, a difficult problem because of the large space of possible graph structures. We use a greedy hill-climbing approach to structure learning, and start by initializing our AND/OR graph structure using a random initialization. To help constrain the space of possible graph structures, we constrain each node to have at most λ_{child} children and λ_{parent} parents. By constraining the number of children a node can have, we help regularize our graph structures so that we select the most important kernels. By constraining the number of parents a node can have, we prevent kernels from appearing in large numbers of nodes in the graph, allowing our structure to consider different subsets of kernels.

After initializing our graph structure G, we select a random node V_i in the graph and consider the following set of moves:

- **Add operation.** We add a node from the layer below V_i to be a child of this node, which corresponds to incrementing T_{V_i} with an additional node. This move

Event Class	Chance	Tang *et al.* [19]	Greedy	Average	MKL L1	MKL L2	Our Method
1. Attempting a board trick	1.18%	15.44%	22.50%	22.81%	21.66%	**23.07%**	22.87%
2. Feeding an animal	1.06%	3.55%	5.60%	6.11%	5.05%	6.13%	**7.75%**
3. Landing a fish	0.89%	14.02%	19.38%	25.43%	27.63%	25.67%	**30.31%**
4. Wedding ceremony	0.86%	15.09%	34.25%	31.78%	37.43%	31.89%	**38.12%**
5. Working on a woodworking project	0.93%	8.17%	21.42%	**24.96%**	18.57%	24.79%	21.50%
Mean AP	0.98%	11.25%	20.63%	22.22%	22.07%	22.31%	**24.11%**

Table 1. Average Precision (AP) values for the MED DEV-T dataset.

Event Class	Chance	Tang *et al.* [19]	Greedy	Average	MKL L1	MKL L2	Our Method
6. Birthday party	0.54%	4.38%	14.83%	8.82%	9.31%	8.84%	**15.97%**
7. Changing a vehicle tire	0.35%	0.92%	27.00%	24.37%	22.66%	25.04%	**31.92%**
8. Flash mob gathering	0.42%	15.29%	42.84%	43.64%	41.63%	44.02%	**44.11%**
9. Getting a vehicle unstuck	0.26%	2.04%	12.01%	**17.89%**	12.47%	17.68%	16.32%
10. Grooming an animal	0.25%	0.74%	6.52%	6.41%	**11.38%**	6.43%	11.00%
11. Making a sandwich	0.43%	0.84%	11.12%	14.91%	13.14%	**15.36%**	14.29%
12. Parade	0.58%	4.03%	12.75%	10.52%	17.64%	10.43%	**18.53%**
13. Parkour	0.32%	3.04%	25.62%	21.12%	21.01%	21.59%	**26.10%**
14. Repairing an appliance	0.27%	10.88%	26.52%	23.15%	24.01%	23.22%	**27.03%**
15. Working on a sewing project	0.26%	5.48%	6.50%	**12.50%**	11.99%	12.36%	12.49%
Mean AP	0.37%	4.77%	18.57%	18.33%	18.52%	18.50%	**21.78%**

Table 2. Average Precision (AP) values for the MED DEV-O dataset.

is not permitted if it generates a structure that violates our constraints on λ_{child} and λ_{parent}.

- **Remove operation.** We remove a child node from V_i, which corresponds to removing a node from T_{V_i}. This move is not permitted if T_{V_i} only contains one node.

- **Swap operation.** We swap one of the child nodes from V_i with one of the child nodes from another node V_j, where V_j is in the same layer as V_i. This corresponds to swapping a node from T_{V_i} for a node in T_{V_j}.

Considering each of these moves provides us with a set of potential graph structures $\{G_1, G_2, \ldots, G_k\}$. For each potential graph structure G_i, we perform inference to find the best configuration. Then, we compute the structure score $Struct(G_i)$ using the following equation:

$$Struct(G_i) = S(G_i(V_{root})) - \lambda_{struct}|G_i(V_{root})| \quad (6)$$

where $G_i(V_{root})$ corresponds to the root node of the potential graph structure G_i. This score is a combination of the score of the root node, combined with a regularization on the number of LEAF nodes selected by the root node to prevent overly complex combinations. This regularization is similar to Bayesian scores used in structure learning for graphical models, such as the Bayesian Information Criterion. Using this score, we select the structure with the best score, and update our structure to this new structure. We then repeat this move-making process until convergence or a maximum number of iterations has been reached. Figure 4 illustrates the whole procedure.

Efficiency. Much of the computation in the bottom-up processing stage of our inference procedure can be re-used for each of the potential structures. Any subtree that remains unchanged by the graph moves does not need to be re-computed, as the optimal bottom-up configuration will remain the same. In practice, we use a hash table to keep track of the scores for all leaf node combinations that have been computed.

6. Experiments

We perform experiments on the 2011 TRECVID Multimedia Event Detection (MED) dataset [17], which consists of a collection of Internet videos collected by the Linguistic Data Consortium from various Internet video hosting sites. There are 15 events split into two sets, the DEV-T set with 5 events, and the DEV-O set with 10 events. There are approximately 150 training videos for each event, and in the two testing sets for DEV-T and DEV-O, we are given large databases of videos that consist of both the events in the set as well as null videos that correspond to no event. There are a total of 10,723 videos in the DEV-T test set, and 32,061 videos in the DEV-O test set. Similar to [19], we consider results for the two sets separately, as it is stated that there may be unidentified positive videos of events from the DEV-T set in the DEV-O test set, and vice versa.

6.1. Implementation Details

We extracted 13 types of image features using code from [22]. These features include variants of GIST [16], HOG [4], SIFT [13], LBP [15] descriptors, and other de-

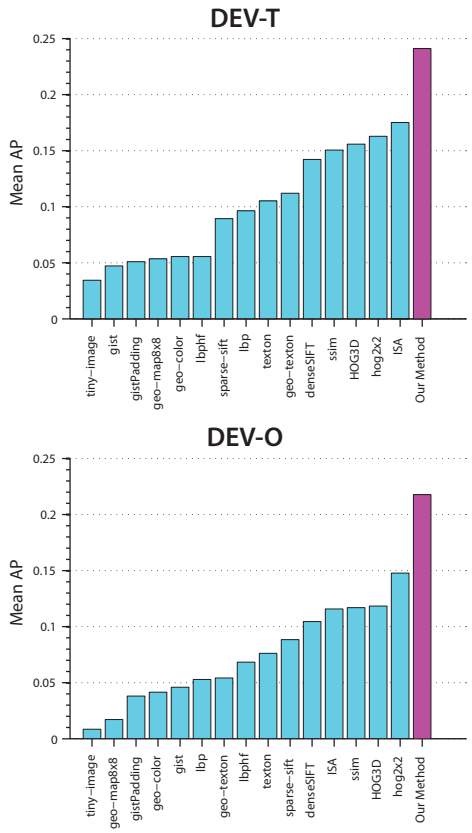

Figure 5. Performance of individual features versus our method for the DEV-T and DEV-O datasets.

Dataset	Layer 2	Layer 3	Layer 4	Layer 5
DEV-T	9.91%	12.80%	18.27%	24.11%
DEV-O	7.30%	11.26%	15.69%	21.78%

Table 3. Average Precision (AP) values for datasets using graph structures with different numbers of layers.

scriptors capturing texture, color, and geometry. We computed these image features on sampled frames every 4 seconds, and component-wise averaged each of them to obtain our video feature. We also extracted HOG3D [10] and ISA [12] video features, and computed video-level histograms for both. For all features, we used the Histogram Intersection Kernel for our kernel matrices, normalized using spherical normalization [11], as this kernel provided us with the best individual feature results. For all methods that define a combination of kernels, we train an SVM over the kernel combination, and cross-validate to determine the C parameter. For each class, we train a one-versus-all classifier and compute average precision on the test set.

For our method, we chose $\lambda_{child} = 7$ and $\lambda_{parent} = 5$ based on the number of kernels we had. The value of λ_{struct} was determined by cross-validation. From our experiments, we found that the choice of these parameters did not make

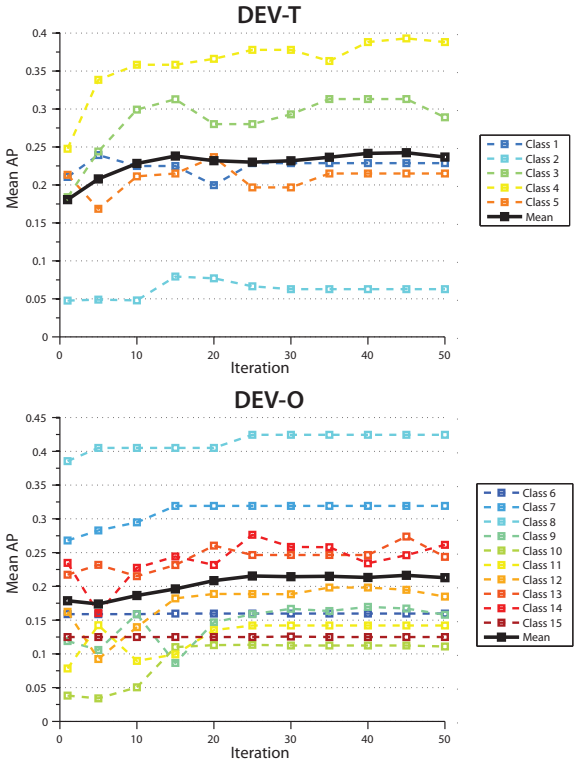

Figure 6. Performance of our method over iterations of structure learning for each class in the DEV-T and DEV-O datasets. We also show the averaged performance over all classes in each plot. Class numbers are given in Tables 1 and 2.

a big difference after a certain threshold, as the best performance is typically obtained using a small subset of the kernels anyways. To constrain our search space, we considered 5-layer AND/OR graphs (see Figure 8), with alternating layers of AND nodes and OR nodes. To help alleviate the problem of local optima in our structure search procedure, we considered multiple random initializations, and selected the graph structure whose configuration provided us with the lowest energy.

6.2. Comparisons

Greedy. This method iteratively selects the best individual performing feature through cross-validation, and combines this feature with all previously selected features using kernel averaging. This combination is evaluated using cross-validation, and the algorithm stops when the newest combination decreases the cross-validated performance.

Average. This method averages the kernels for all the features, which has been show to be very competitive to more complicated methods such as MKL [5].

MKL. This method is Multiple Kernel Learning, which solves a joint optimization problem that simultaneously selects the weights to combine kernels with, as well as the

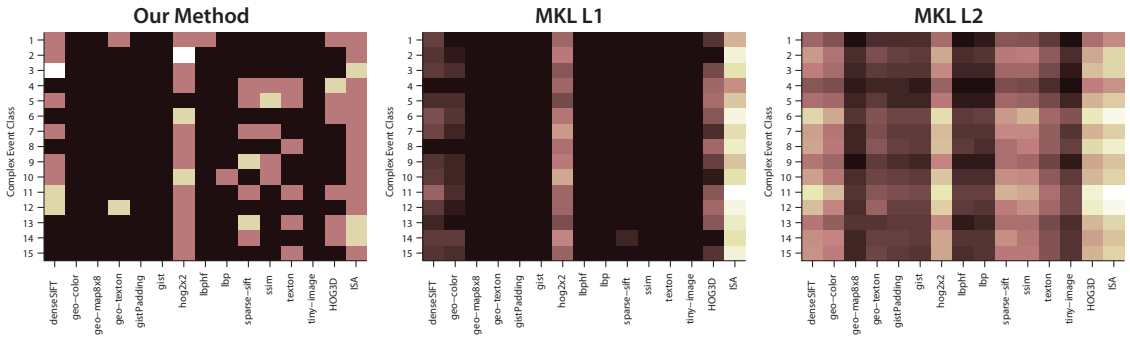

Figure 7. Visualizations of the feature combinations learned by various methods for each of the complex event classes. The matrix visualizes the weights learned for each of the features. Lighter colors correspond to higher weights. Class numbers are given in Tables 1 and 2.

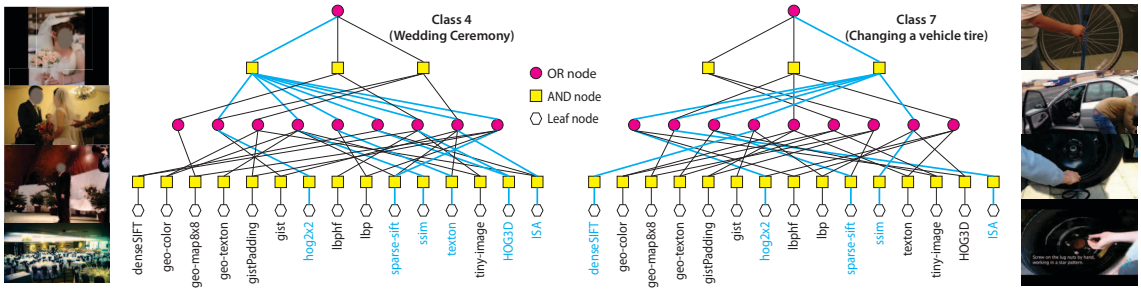

Figure 8. Visualizations of AND/OR graph structures that are learned by our method for the "Wedding ceremony" and "Changing a vehicle tire" classes. The blue edges indicate the optimal configuration chosen after inference. We show sample frames for each event on the side.

weights of the classifier. The L1 and L2 variants use L1 and L2 normalization on the kernel weights. We use code provided by [20]. There are many complicated variants of MKL, but as noted in [6], the difference in terms of accuracy between methods is usually quite small.

Tang *et al.* [19] This method uses temporal structure for complex event recognition with only HOG3D features. This method is shown only to give an idea of the performance of previous works on the same split of the dataset.

6.3. Results

Our results for the DEV-T dataset are given in Table 1, and DEV-O dataset in Table 2. Our method outperforms all other methods in mean AP. Note that a 2-3% increase in mean AP on these datasets is significant, as the test set consists of a large number of videos. As observed in [5], the performance of kernel averaging is comparable to MKL. Although kernel averaging is a special instance of our method where an AND node combines all LEAF nodes, our method sometimes performs worse than averaging. This is because we place several forms of regularization on our model including the λ_{child} and λ_{parent} parameters so that our method prefers simpler kernel combinations, and constrains the space of AND/OR graphs we must search over. However, it is possible to search an even larger space of AND/OR graph structures that includes kernel averaging, and that would help improve performance further.

We show results comparing our method to the individual features in Figure 5. In this work, we considered several image and video features, some of which work better than others. Note that we could extract many additional features, such as audio features, and get an additional boost in performance. However, our goal is to illustrate the benefits of our method for combining features, and not to try to exhaust all types of features for the best possible performance.

The performance of our method over iterations of structure learning is plotted in Figure 6, where we see that it seems to converge to a local optima by iteration 50. The convergence will likely be much slower if we considered more complicated graph structures or additional types of moves. However, we found that the moves we proposed gives a good balance between exploring the full space of potential structures and efficiency. In both datasets, we obtain a 4-5% gain in mean AP over all classes after our structure learning procedure ends. Note that the performance of the initial graph structures are decent, as we perform inference on these structures to obtain their optimal configurations. In Table 3, we also show the performance of graph structures with different numbers of layers. As we increase the number of layers, we generally see an increase in performance.

In Figure 7, we illustrate the feature combinations chosen by our method and the two variants of MKL. Not

surprisingly, MKL L2 is unable to do feature selection, and non-zero weights are used to combine all features. More interestingly, MKL L1 extensively favors the denseSIFT, geo-color, hog2x2, HOG3D, and ISA features for all classes. Our method considers additional features such as geo-texton, lbphf, lbp, sparse-sift, ssim, and texton. From Figure 5, we see that many of these features don't perform well individually. However, because our method considers features independently in a hierarchical setting, it allows us to discover complementary features otherwise missed by MKL L1. We visualize two of the learned graph structures and configurations in Figure 8. We can see how the hierarchical structure of the graphs allow us to consider different features in conjunction with others. The "Changing a vehicle tire" graph visualization shows how our method prefers SIFT image features for this class, possibly because the presence of tires is very indicative. Note that our method is also able to do implicit kernel weighting, as seen in the graph visualization for "Wedding ceremony", where the HOG3D feature is deemed important and combined twice.

7. Conclusion

In conclusion, we have presented a method for combining features that incorporates our intuitions for how features should be combined. Our method uses an AND/OR graph to represent possible feature combinations, and automatically learns the structure of the graph. Using the AND/OR graph structure, our feature combination method is able to be selective of features, consider different subsets of features in a hierarchical manner, and achieve convincing results on the 2011 TRECVID MED dataset [17].

There are many possible directions for future work. We placed several restrictions on our AND/OR graphs to decrease the number of potential structures we had to consider. Designing efficient methods to utilize additional layers and nodes with non-linear behavior could be a possible direction. In addition, it would be interesting to draw connections between our method and objectives that are optimized by kernel combination techniques such as MKL.

Acknowledgments. We thank Andrej Karpathy, Guido Pusiol, and Vignesh Ramanathan for helpful comments. This research is partially supported by an ONR MURI grant, the DARPA Mind's Eye grant, and a NSF GRFP under grant no. DGE-114747 (to K.T.).

References

[1] F. Bach. Exploring large feature spaces with hierarchical multiple kernel learning. In *NIPS*, 2008. 2

[2] H. Chen, Z. Xu, Z. Liu, and S. C. Zhu. Composite templates for cloth modeling and sketching. In *CVPR*, 2006. 2, 4

[3] Y. Chen, L. Zhu, C. Lin, A. L. Yuille, and H. Zhang. Rapid inference on a novel and/or graph for object detection, segmentation and parsing. In *NIPS*, 2007. 2, 4

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 5

[5] P. V. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *ICCV*, 2009. 2, 6, 7

[6] M. Gönen and E. Alpaydin. Multiple kernel learning algorithms. *JMLR*, 2011. 2, 7

[7] A. Gupta, P. Srinivasan, J. Shi, and L. S. Davis. Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In *CVPR*, 2009. 2

[8] S. J. Hwang, K. Grauman, and F. Sha. Semantic kernel forests from multiple taxonomies. In *NIPS*, 2012. 2

[9] H. Izadinia and M. Shah. Recognizing complex events using large margin joint low-level event model. In *ECCV*, 2012. 2

[10] A. Kläser, M. Marszałek, and C. Schmid. A spatio-temporal descriptor based on 3d-gradients. In *BMVC*, 2008. 6

[11] M. Kloft, U. Brefeld, S. Sonnenburg, and A. Zien. lp-norm multiple kernel learning. *JMLR*, 12:953–997, 2011. 6

[12] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 2011. 6

[13] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 5

[14] P. Natarajan, S. Wu, S. N. P. Vitaladevuni, X. Zhuang, S. Tsakalidis, U. Park, R. Prasad, and P. Natarajan. Multimodal feature fusion for robust event detection in web videos. In *CVPR*, 2012. 1, 2

[15] T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE TPAMI*, 2002. 5

[16] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 2001. 5

[17] P. Over et al. Trecvid 2011 – an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *TRECVID 2011*, 2011. 1, 2, 5, 8

[18] A. Tamrakar, S. Ali, Q. Yu, J. Liu, O. Javed, A. Divakaran, H. Cheng, and H. S. Sawhney. Evaluation of low-level features and their combinations for complex event detection in open source videos. In *CVPR*, 2012. 2

[19] K. Tang, L. Fei-Fei, and D. Koller. Learning latent temporal structure for complex event detection. In *CVPR*, 2012. 5, 7

[20] M. Varma and D. Ray. Learning the discriminative power-invariance trade-off. In *ICCV*, 2007. 2, 7

[21] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *ICCV*, 2009. 2

[22] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 5

[23] Y. Yang and M. Shah. Complex events detection using data-driven concepts. In *ECCV*, 2012. 2

[24] B. Yao and L. Fei-Fei. Grouplet: A structured image representation for recognizing human and object interactions. In *CVPR*, 2010. 2

[25] L. Zhu, Y. Chen, Y. Lu, C. Lin, and A. L. Yuille. Max margin and/or graph learning for parsing the human body. In *CVPR*, 2008. 2