A generic model to compose vision modules for holistic scene understanding

Congcong Li*,[1], Adarsh Kowdle*,[1], Ashutosh Saxena[2], Tsuhan Chen[1]

[1] School of Electrical & Computer Engineering, Cornell University
[2] Department of Computer Science, Cornell University
{cl758,apk64}@cornell.edu,asaxena@cs.cornell.edu,tsuhan@ece.cornell.edu

Abstract. The problem of holistic scene understanding involves many vision tasks such as depth estimation, scene categorization, event categorization, etc. Each of these tasks explores some aspects of the scene but, these tasks are related in that, they represent attributes of the same scene. An intuition is that one task can provide meaningful attributes to aid the learning process of another task.
In this work, we propose a generic model (together with learning and inference techniques) for connecting different vision tasks in the form of a 2-layer cascade. Our model considers the first layer as a hidden layer, where the latent variables are inferred by feedback from the second layer. The feedback mechanism allows the first layer classifiers to focus on more important image modes, and draws their output towards "attributes" rather than the original "labels". Our model also automatically discovers sparse connections between the learned attributes on the first layer and the target task on the second layer. Note that in our model, the same vision tasks can act as attribute learners as well as target tasks, while being set up on different layers. In extensive experiments, we show that the *same* proposed model improves the performance in *all* the tasks we consider: single image depth estimation, scene categorization, saliency detection and event categorization.

1 Introduction

One of the primary goals in computer vision is holistic scene understanding, which involves many sub-tasks, such as depth estimation, scene categorization, saliency detection, object detection, event categorization, etc. Each of these tasks explains some aspects of a particular scene. To fully understand a scene, the goal is to solve for each of these tasks. However, these tasks are related in that they represent some attributes of the same scene and thus can aid each other, as shown in many recent works, e.g., [1–4]. These methods, however, either tend to be ad-hoc (where a manually engineered rule is used to combine the tasks) or would need considerable attention to construct (probabilistic) models to combine them, based on the insights of the researchers into the problem of scene understanding. We would like to develop a more generic model that automatically discovers the connections or relationships between different tasks. A recent

* indicates equal contribution

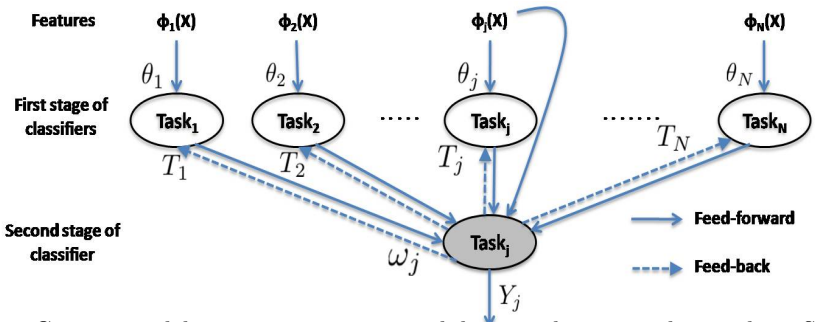


Fig. 1. Generic model to compose vision modules to aid scene understanding. Sparse cascaded classification model with feedback, where the decision from the second layer is fed back to the previous layer to learn useful attributes thus to help improve final performance. (Each $Task_i$ in the first layer is parametrized by θ_i and its output is T_i. The output from the second stage classifier for $Task_j$ parametrized by ω_j, is Y_j)

method called Cascaded Classification Models (CCM) by Heitz et. al. [5] attempts to do so by repeating instantiations of the classifiers for different tasks in a cascade. The classifiers on each layer are trained to the groundtruth "labels" of corresponding tasks. The restriction of training each layer classifiers towards "labels" can easily cause the overfitting problem. In this work, we propose a 2-layer CCM model with a hidden intermediate layer, which automatically learns meaningful attributes for optimizing the target task on the second layer.

The idea of learning attributes towards a particular target task relates our work with some recent research about learning visual attributes, e.g. [6–9]. The prior works learn attributes to describe an image, mainly for object recognition. In this work, we look at scene understanding where we try to describe a scene by various explicit attributes, such as the type of scene, the event going on in the scene, or other implicit attributes such as the depths of different regions, the salient regions in the scene, etc. These attributes, provide specific description of the scene, which if learned by the tasks mentioned earlier can serve as valuable inputs to the target task. For example, it is intuitive that a "mountain-like" attribute plays an important role in identifying a "snowboarding" event and a "depth in the top region" attribute helps identify a "tall-building" scene. Based on this analysis, we propose a generic model in the form of a 2-layer cascade, where we have various tasks on the first layer and repeat the particular target task on the second layer, as shown in Figure 1. Note that in our model, the same vision tasks can act as attribute learners as well as target tasks, while being set up on different layers. This is different from the prior works, which need to introduce extra tasks (e.g. color classification, shape classification, etc.) for attribute learning.

Our approach is as follows. We have classifiers for N tasks on the first layer, and our goal is to improve the performance of a particular task j on the second layer. Our learning algorithm tries to maximize the conditional log-likelihood of the final output. The goal of the second-layer classifier is to achieve the best class labels while the first-layer classifiers are not trained to do so. The target outputs for the first-layer classifiers are latent variables, which are initialized with the groundtruth labels of the corresponding tasks and are suitably inferred by the feedback from the second-layer classifier in later iterations during the training

phase. The feedback passes the first layer information about what regular modes they should focus on to describe the scene. Implicitly, the feedback draws the first-layer outputs towards "attributes" rather than "labels". For example, we observe that the feedback allows a "bocce" event image to have high scores for both "bocce" and "croquet" categories on the first-layer, indicating the existence of both attributes "bocce-like" and "croquet-like", which is consistent with our intuition that these two events share many common characteristics. Furthermore, to connect a large number of first-layer tasks with the second-layer task, we also want the connections to be sparse, i.e., the target task is connected to only a few meaningful attributes. In this work, we use Laplace prior to incorporate sparsity into our model, which figures out the connections automatically.

Our algorithm has other interesting properties that are of interest to many vision problems. Typically, each of these tasks have their own independent training datasets, i.e., for an image, the labels for all the tasks may not be available. We will show in detail how our model can intrinsically handle such a situation, and thus scales well even with the *heterogeneous* datasets.

The contributions of our algorithm are as follows:

1. Our algorithm incorporates feedback into the cascaded model, which helps to train the first-layer tasks towards attributes for the benefit of the target task on the second layer.
2. Our algorithm incorporates sparsity by using the Laplace prior in order to learn how the learnt attributes contribute to the target task.
3. Our algorithm handles *heterogeneous* datasets.
4. We consider four different vision tasks: single image depth estimation, scene categorization, saliency detection, and event categorization. In our extensive experiments we show that using the *same* algorithm we achieve improvements in each of these tasks over state-of-the-art classifiers.

2 Related work

The importance of combining different tasks together (for "holistic scene understanding") has been highlighted by a number of recent works [2–4, 10–14]. For example, Sudderth et. al. [4] incorporate object detection to help 3D structure estimation. Hoiem et. al. [3] combine multiple aspects of the scene like object boundaries and surface orientations towards 3D scene interpretation. Li et. al. [2, 14] propose hierarchical graphical models to integrate multiple related tasks such as detection, classification, etc. With well-designed models, these works outperform independent methods, which shows the potential of sharing information between multiple tasks. However, the design of the model structure requires considerable attention to each task and insight into modeling the connection between tasks. This can limit the generality of introducing more tasks into the model, or applying it to other tasks.

On the other extreme, generic approaches have been developed in machine learning community to combine many classifiers to achieve a better overall performance. Boosting [15] is one among these methods, where many weak learners are combined to obtain a more accurate classifier. One highly relevant work to ours is by Heitz et. al. [5], where a generic approach called Cascaded Classifier

Models (CCM) is proposed for connecting multiple tasks in scene understanding. CCM method considers each individual classifier as a "black-box" and therefore it is compatible to different classifiers. However, CCM is limited because it sequentially optimizes each layer classifiers independently instead of optimizing the final output likelihood conditioned on the observations. This independent optimization scheme can easily suffer from the overfitting problem, and this becomes critical when combining a large number of tasks. In our work, we propose a feedback mechanism to allow the communication between layers in order to focus on more important modes on the first layer. We also introduce sparsity to find a limited number of links between layers in order to avoid overfitting.

Another group of works related to ours are those on "visual attribute learning", e.g. [6–9]. Most of these works introduce visual attributes to aid object recognition, which are defined as properties that can be shared across object categories such as shape, material, etc. In our work, we link various attributes towards holistic scene understanding. Our attributes can be explicit ("it is a more mountain-like scene rather than a coast-like scene") or implicit ("it has low depth in the upper region of the scene"), etc. Moreover, our model is generic for any particular sub-task in the area of scene understanding. The proposed model is scalable in that, we can add any related tasks to the first-layer to aid the target task. Our feedback mechanism results in learning attributes for the particular target task instead of learning towards the original category labels on the first-layer. In our model the same vision tasks can act as attribute learners as well as target tasks, while put on different layers.

3 Our Approach

Our model was designed to handle the following properties:

1. Ability to compose the individual classifiers without information on inner workings of each classifier. I.e., use state-of-the-art classifier as "black-box".
2. Learning should be feedback enabled for each classifier. i.e., each classifier should get hints on which modes to focus its attention on. First-layer classifiers can learn to output meaningful attributes in favor of the target task.
3. Ability to handle heterogeneous datasets, i.e., each task could come with its separate labeled data. This is a very powerful advantage because most vision datasets are such.
4. Scale with large number of tasks. i.e., the same model should work if we increase the number from 4 (in this paper) to say, 10. Our model should automatically decide how to compose the tasks sparsely.
5. Tractable inference. The inference in our model should be tractable and its complexity should be no more than constant times the complexity of inference in the original classifiers.

Notation: Our cascade is built in two layers, as shown in Figure 1. We consider that there are N tasks in first layer that are being combined in this cascaded model denoted as Task_i where $i = 1, 2, \ldots, N$. Our goal is to combine them for the task say Task_j, at the output of the second layer in the cascade. Let the dataset for j^{th} task be Γ_j, which consists of a labeled pair $\{X^{(k)}, Y_j^{(k)}\}$, where k indices over all the datapoints in the training set.

In the first layer, the classifier for Task$_i$ takes image features $\Phi_i(X)$ as input and gives output T_i using parameters θ_i. In the second layer, Y_j is the output for the j^{th} task which uses the original features X_j as well as all the outputs from the first layer as input, with the parameters ω_j. For brevity, we will use Θ to indicate the set $\{\theta_1, \ldots, \theta_N\}$.

Model: Our goal is to learn the parameters by maximizing the conditional log-likelihood $P(Y_j|X)$ over all the data-points in the training-set Γ_j for task j. In order to scale well we use a sparse coding [3] of the inputs to the second layer.

$$\underset{\omega_j,\Theta}{\text{maximize}} \quad \log \prod_{k \in \Gamma_j} P(Y_j^{(k)}|X^{(k)}, \omega_j, \Theta) P(\omega_j) P(\Theta) \tag{1}$$

Here, $P(\omega_j)$ and $P(\Theta)$ are the terms that enforce sparsity. Note that the parameter set ω_j is typically extremely high-dimensional (and increases with the number of tasks) because the second layer classifiers take as input the original features as well as outputs of all previous layers. The learning problem becomes ill-conditioned, and therefore in most prior works, the solution is to manually decide how to connect the pieces together using intuition and insight.

In our work, we want our model to select only a few non-zero weights, i.e., only a few non-zero entrees in ω_j. We do by using Laplace prior $P(\omega_j) \propto \exp(-\lambda||\omega_j||_1)$. This prior is known to enforce sparsity in the parameters [17].

$P(\Theta) = P(\theta_1, \theta_2, \ldots, \theta_N)$ on the other hand, is a prior over the parameters of the first level of classifiers, which only use the image features as inputs. Since we consider these classifiers to be independent state-of-art black-box classifiers, the prior $P(\theta_1, \theta_2, \ldots, \theta_N) = \prod_{i=1}^{N} P(\theta_i)$. Since we want to consider the individual classifiers as "black-boxes," we would use the same prior as used by the state-of-art classifiers (uniform prior in the case of none); typically the image features selected for these tasks are all relevant (e.g., they were carefully hand-designed).

Now we present our derivation for the training procedure. First we note that during training, Y_j are observed (because the ground-truth labels are available), however $T_i \ \forall i$ (output of layer 1 and input to layer 2) are hidden. This makes training of each classifier hard. Heitz et. al. [5] assumed that each layer is *independent* and that each layer produces the best output independently (regardless of the later layers); this is not optimal because the first layer classifiers could be focusing on error modes that are not necessary for final output for task j.

In this work, we formulate the cascaded model as a probabilistic model which allows for a better learning algorithm in optimizing the final output. Using Figure 1, we write Eqn 1 as,

$$\underset{\omega_j,\Theta}{\text{maximize}} \quad \sum_{k \in \Gamma_j} \log \left(\sum_{T_1^{(k)}, \ldots, T_N^{(k)}} P(Y_j^{(k)}, T_1^{(k)}, \ldots, T_N^{(k)}|X^{(k)}, \omega_j, \Theta) P(\omega_j) \right) \Leftrightarrow$$

$$\underset{\omega_j,\Theta}{\text{maximize}} \sum_{k \in \Gamma_j} \log \left(\sum_{T_1^{(k)}, \ldots, T_N^{(k)}} \left(P(Y_j^{(k)}|X^{(k)}, T_1^{(k)}, \ldots, T_N^{(k)}, \omega_j) P(\omega_j) \prod_{i=1}^{N} P(T_i^{(k)}|X^{(k)}, \theta_i) \right) \right) \tag{2}$$

[3] Sparse coding is a technique used for finding concise, slightly higher-level representations of inputs, and was applied in a an unsupervised setting in [16].

The summation inside the log makes it difficult to learn the parameters. Motivated by the Expectation Maximization [18] algorithm, we use algorithm where we first fix the latent variables and learn the parameters in the first step (*Parameter learning*), and then, estimate the latent variables (*Latent variable estimation*) with the parameters fixed. While this algorithm is not guaranteed to converge to the "global" maxima, in practice, we found it gives good results, typically after 5 - 10 iterations.

***Parameter learning*:** In this step, we assume that the latent variables are known (from the Latent-variable-estimation step).[4] The summation inside the log in Eqn 2 goes away, and the maximization problem then breaks into two separate problems as follows.

$$\forall i \in \{1, \ldots, N\}, \quad \underset{\theta_i}{\text{maximize}} \sum_{k \in \Gamma_j} \log P(T_i^{(k)} | X^{(k)}, \theta_i) \tag{3}$$

$$\underset{\omega_j}{\text{maximize}} \sum_{k \in \Gamma_j} \log P(Y_j^{(k)} | X^{(k)}, T_1^{(k)}, \ldots, T_N^{(k)}, \omega_j) P(\omega_j) \tag{4}$$

Each of the maximization problem in Eqn 3 is precisely the learning problem of the "blackbox classifier," and we would use the learning method provided by the individual blackbox classifier. Note that in this part, the individual classifiers need not have a probabilistic interpretation (e.g., SVM); they just need to be able to train their parameters given a training set (using whatever algorithm).

The maximization problem in Eqn 4 is also an instantiation of the original learning problem, but with more inputs. There are several methods to approach this problem depending on the situation as follows:

1. Case 1: No insight into the vision problem, and no probabilistic interpretation of the original classifier for task j is available.
   In this case, we will append all the outputs $T_1, \ldots, T_N$ to the original feature vectors $X^{(k)}$, and make a variational approximation on the output of the classifier for task j (i.e., approximating it as a Gaussian, [19]) to get:
   $$\underset{\omega_j}{\text{minimize}} \sum_{k \in \Gamma_j} \left(\left|\left| Y_j - \omega_j^T [X^{(k)}, T_1, \ldots T_N] \right|\right|_2^2 + \lambda \left|\omega_j\right| \right) \tag{5}$$
2. Case 2: No insight into the vision problem, and probabilistic interpretation of the original classifier is available.
   In this case, we just append all the outputs $T_1, \ldots, T_N$ to the original feature vectors $X^{(k)}$, and solve problem Eqn 4 using exactly the same method as provided by the original classifier.
3. Case 3: Insight into the vision problem is available.
   In this case, instead of feeding the outputs of the previous layer as appended features, we would use our insights into the problem to properly model $P(Y_i | X, T_1, \ldots, T_N)$.

In this paper, we show that even with Case 1, we get uniformly better results across tasks. We believe that, if for a particular problem, one has insights into

[4] We initialize the model by setting the latent variables T_i $\forall i$, to the ground truth for the respective tasks.

the vision problem, one could use case 3 and achieve even better results than what we present here.

***Latent variable estimation*:** In this step, we assume that parameters ω_j and θ_i are known. We now compute $T_1^{(k)}, \ldots, T_N^{(k)}, \forall k \in \Gamma_j$. Using Eqn 2, we get,

$$\begin{aligned} \forall k \in \Gamma_j, \quad & \underset{T_1,\ldots,T_N}{\text{maximize}} \quad \log P(Y_j^{(k)}, T_1, \ldots, T_N | X^{(k)}; \omega_j, \theta_1, \ldots, \theta_N) \\ \Leftrightarrow \forall k \in \Gamma_j \quad & \underset{T_1,\ldots,T_N}{\text{maximize}} \quad \log P(Y_j^{(k)} | X^{(k)}, T_1, \ldots, T_N) + \sum_{i=1}^{N} \log P(T_i | X^{(k)}; \theta_i) \end{aligned} \tag{6}$$

As discussed in the previous section on *Parameter estimation*, we can make a variational approximation to these probabilities if the characterization of the individual black-box classifiers is not available in a probabilistic form. (If the probabilistic characterization is available, we would use that characterization.)

The feed-back step gives optimal values for the latent variables which allows us to obtain optimal performance at the second stage. There is an added advantage of the feed-back step which enables us to work with heterogeneous datasets. Using this step, we automatically get the estimates of the latent variables for Task_i for the dataset $\Gamma_j, (i \neq j)$ that do not have all the labels when we started the algorithm. These optimal latent variablesare used in the parameter learning step again and the alternating algorithm iterates to convergence.

Inference. During inference, the parameters are already estimated, and the inference (i.e., maximizing the conditional log likelihood $P(Y_j|X)$ for a given image X) corresponds to performing inference over first layer (using existing inference techniques in the black-box classifiers), followed by inference in the second layer, again using the existing inference techniques in the black-box classifiers with the extra features, i.e., previous stage's outputs. Thus the total time required for inference in our model is no longer than the total time of the inference for $(N+1)$ individual classifiers, where N is the number of tasks on the first layer. Since our structure allows paralleled inference for different tasks on the first layer, the inference time can then decrease to the maximum inference time for individual classifiers plus the inference time for *task* j on the second layer.

4 Implementation

To test our algorithm, we consider four different vision tasks (i.e., $N = 4$): event categorization, depth estimation, scene categorization and saliency detection. In this section, we describe our implementation for each of our classifiers.

Event Categorization: For event categorization, our goal is to classify an image into one of the 8 sports events as defined in [14]: bocce, badminton, polo, rowing, snowboarding, croquet, sailing and rock-climbing. We define the output of an event classifier to be an 8-dimensional vector with each element representing the log-odds score for each category. For evaluation, we compute the accuracy of assigning the correct event label to an image. For an event categorization module, we implement a multi-class logistic classifier. In the first-layer event classifier, we use a 51-dimensional feature vector as input, including the top 30 PCA projections of the 512-dimensional GIST features [20], the 12-dimension global color features (the mean and variance of RGB and YCrCb channels over

the entire image), the 8-dimension object detection features (the absent indicator and the detected number of the four object classes: boat, horse, person, and car), and a bias term. We apply the latent-svm method [21] for object detection, and we train the detectors with the PASCAL 2008 dataset [22], which is not used for any training/testing for any of the four tasks we consider here.

Depth Estimation: For single image depth estimation, the goal is to estimate the depth of each point in the image. The output of a depth estimation module is therefore a real-value for each point in an image, evaluated by measuring the root mean squared error between the estimated depth and the ground truth laser depth same as in Make3D range image dataset [1]. We use superpixel-level features extracted from the image as described in [1] as input for depth estimation. The features include the texture, color and texture gradients extracted by convolving the image with Laws' masks and computing the energy and Kurtosis over the superpixel, and the superpixel shape features. For the depth estimation module, we use a linear regression to help estimate the depth of every point in the image. Note that though we are using a simpler version (i.e., without Markov Random Field), our model gives better results (e.g., better than Make3D that uses MRF formulation).

Scene Categorization: For scene categorization, our goal is to classify an image into one of the 8 categories defined in [23]: tall building, inside city, street, highway, coast, opencountry, mountain and forest. We define the output of a scene classifier to be a 8-dimensional vector with each element representing the score for each category. We evaluate the performance by measuring the overall accuracy of assigning the correct scene label to an image on MIT outdoor scene dataset [23]. For the first-layer scene classifier, we use a RBF-Kernel SVM classifier, as used in [24], to classify an image into one of the scene categories. We use a 512-dimensional GIST features [23] as input for the first-layer classifier.

Saliency Detection For saliency detection, our goal is to classify each point in the image to be either salient or non-salient. Therefore we define the output of a saliency classifier to be a scalar indicating the salient confidence score at each point in the image. To get the final result, we threshold the saliency score to decide whether it is a salient point or not. We use the saliency detection dataset used in [25] for evaluation. For the saliency detection module, we use logistic regression to help estimate the saliency score of every point in the image. In the first layer of the cascaded models, we use a 3-dimensional feature vector based on the Lab color space as input for saliency estimation. The 3-dimensional features are computed using the method described in [25].

We use "Case 1" described in Section 3 for the second layer classifier, where we append the outputs of previous layers to the original features of each task.

5 Experiments and results

We conduct an extensive set of experiments to validate and evaluate the performance of our proposed approach. We first describe the setup of the experiments to help understand our quantitative analysis.

5.1 Experimental Setup

For the four tasks described in Section 4, we use the same proposed model for each task; in each case we optimize the respective task at the second layer of

Table 1. Summary of results for the four vision tasks. Our method improves performance in every single task. Proposed approaches shaded and best model in bold.

Model	Event Categorization (% **Accuracy**)	Depth Estimation (RMS **Error** in m)	Scene Categorization (% **Accuracy**)	Saliency Detection (% **Accuracy**)
Images in testset	1579	400	2688	1000
Chance	12.5	24.6	12.5	50
State-of-art model	73.4 Li et.al. [14]	16.8 (MRF) Saxena et.al. [1]	83.7 Torralba et.al. [24]	82.5 (±0.2) Achanta et.al. [25]
Our base-model	72.0 (±0.8)	18.5 (±0.4)	83.8 (±0.2)	85.5 (±0.2)
All-features-direct	72.6 (±1.5)	16.4 (±0.4)	83.9 (±0.4)	86.2 (±0.2)
CCM (Heitz et.al.)	72.8 (±1.6)	16.2 (±0.4)	83.7 (±0.6)	86.5 (±0.2)
CCM with feedback	73.3 (±1.0)	15.3 (±0.2)	83.8 (±0.6)	87.3 (±0.2)
Sparse-CCM without feedback	73.6 (±1.4)	16.0 (±0.2)	83.9 (±0.2)	86.6 (±0.1)
Sparse-CCM with feedback	**75.3 (±0.6)**	**15.2 (±0.1)**	**85.3 (±0.2)**	**87.3 (±0.1)**

the model. We perform 4-fold cross-validation to evaluate the performance of the event categorization and scene categorization tasks on a set of 1579 images and 2688 images respectively. We perform 2-fold cross validation to evaluate the performance of the depth estimation and the saliency detection tasks on a set of 400 images and 1000 images respectively. (We use the same dataset as the state-of-the-art classifiers for the respective tasks.)

5.2 Results

Table 1 gives a summary of the results for all four tasks. We first descibe the various models we compare. In brief, we highlight that we compare our approach with the state-of-art performance published for each task on that specific dataset and our implementation of the previously proposed cascaded classifier models by Heitz et. al. [5].

1. State-of-art model: We compare with [14] for event categorization, [1] for depth estimation (this uses an MRF framework; nevertheless we show that our final proposed model can outperform this using just linear regression), [24] for scene categorization and [25] for saliency detection [5]. Note that these state-of-the-art methods already use well-designed features/structures, and some [14, 1] also use a lot of context in different forms.
2. Our base-model: This model corresponds to our baseline classifier (i.e., the individual classifiers independently trained). This serves as a base model to evaluate the performance of our algorithm.
3. All-features-direct: This model simply appends together all features used in each task, and builds a separate classifier for each task. Sparsity is used to eliminate overfitting.
4. CCM: This model is our implementation of the cascaded classifier model proposed by Heitz et. al. [5]. (It was verified that the implementation gives similar results.) Note that this model neither has feedback nor sparsity.

   The following three models are the new models we propose:

[5] Most of the prior work on saliency detection are unsupervised. Since we consider supervised classifiers, we use the same features as Achanta et. al. [25] and train a linear regression model using the training set and test it on the test set.

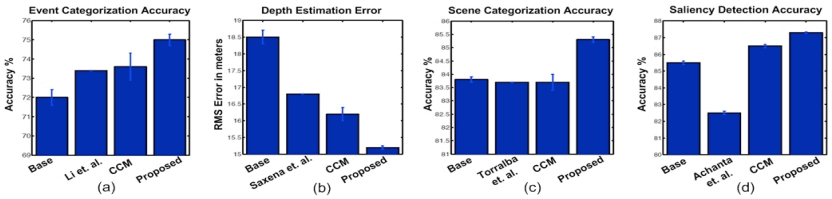


Fig. 2. (a) Event categorization **accuracy**, (b) Depth estimation **error**, (c) Scene categorization **accuracy**, (c) Saliency detection **accuracy**

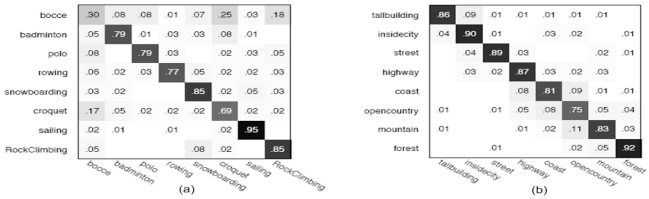


Fig. 3. (a) Event categorization confusion matrix, (b) Scene categorization confusion matrix In each confusion table, the rows represent the ground truth categories while the column represent the estimated categories.

5. CCM with feedback: In this model, our training method based on latent variables is used.
6. Sparse-CCM without feedback: This method uses sparsity, however does not use our training method, i.e., it is equivalent to CCM with the sparsity term.
7. Sparse-CCM with feedback: This is the complete model we propose which incorporates sparsity into the CCM framework along our training method using latent variables.

Table 1 shows that the feedback mechanism and the sparsity property both help improve performance over the CCM model in [5]. Our Sparse-CCM with feedback model outperforms the state-of-art models as well as the original CCM for each task. Furthermore, besides improving the performance, our complete model also provides more stability than all the other methods we compare to (see σ for the results in the table, and error bars in Figure 2). The standard deviations in our experiments show that our improvements in the results are statistically significant.

Categorization Tasks. Figure 3 shows the confusion matrix for event categorization and scene categorization. Note that in our base classifiers we use fewer object detectors when compared to [14], and our detectors are also more coarse, e.g., we do not differentiate rowing boats and sailing boats. Even then our model automatically learns to use the outputs from the different tasks on the first layer efficiently and improves event categorization accuracy. For the task of scene categorization, not only we see improvement in the overall performance, we also achieve either equal or better accuracy for *every* individual category compared to [24] (Figure 3b).

Depth estimation. Figure 4 (first 2 rows) gives a visualization of the performance improvement due to the proposed model. We observe that our model is clearly better than both the CCM model (Heitz et. al.) as well as the base model.

Saliency detection. The last two rows of Figure 4 shows the improvement in saliency detection. Here the cyan color indicates the salient region. Observe in particular that, parts of the sky and ground were incorrectly classified as salient

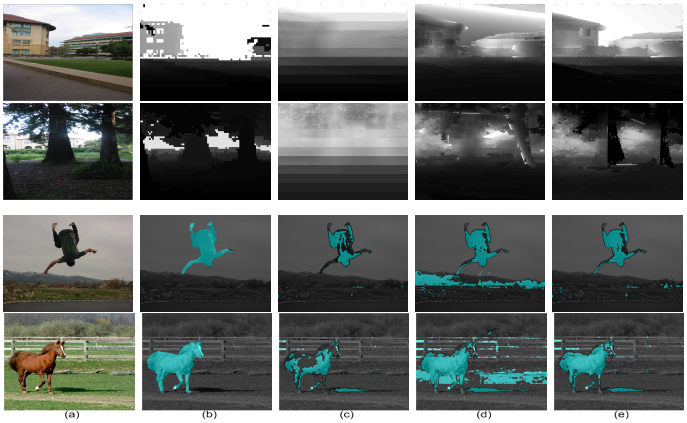


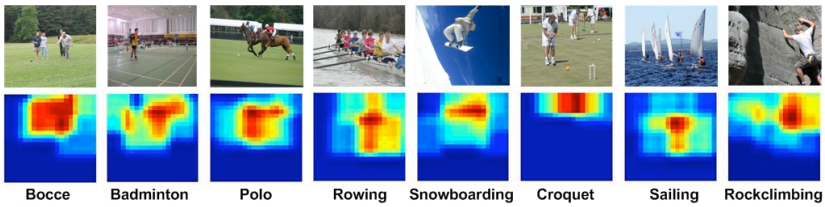
Fig. 4. First two rows show the depth estimation improvement (All are gray scale depth maps). The last two rows show the saliency detection improvement (cyan color indicates salient regions) (a) Image from the dataset; (b) Ground truth; (c) Result from the base-model; (d) Result using the CCM model without feedback (Heitz et. al.); (e) Result from the proposed Sparse-CCM with feedback. (Note that: The depth maps are at the same scale; black means near and white means far).

region in the CCM model however, with the proposed model, the feed-back corrects this to result in a much better saliency detection.

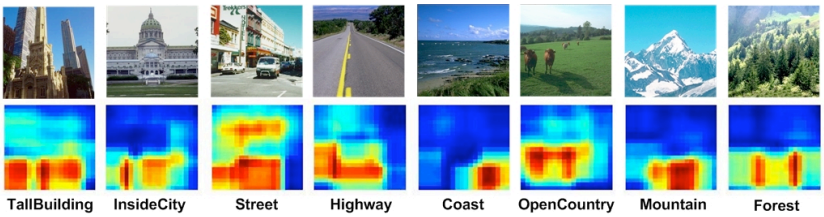
5.3 Discussion

The proposed model for combining classifiers via the latent variable estimation (feedback) step, results in the first-layer classifiers learning meaningful attributes rather than the original target labels. This is different from the model by Heitz et. al. [5] where each layer is trained towards the target labels. For example, in our model, the outputs from the first-layer scene classifier tells whether an image is "open country-like" or "forest-like" or both (thus, it focuses on describing the image modes rather than simply discriminating different classes); the outputs from the first-layer depth estimator gives an attribute of potential depth condition in different regions across the image, etc. Our model then uses the Laplace prior to enforce sparsity in the connections between two layers, resulting in interesting observations about the "attribute"-"target task" relationship.

Non-zero components in the weights ω_j represent the outputs being used in the connections. Figure 5a visualizes the weights given to the depth attributes (first-layer depth outputs) for the task of event categorization. Figure 5b shows them for the task of scene categorization. We see that the depth plays an important role in these tasks. In Figure 5a, we observe that most event categories rely on the middle part of the image, where the main objects of the event often locates. E.g., most of the "polo" images have horses and people in the middle of the image while many "snowboarding" images have people jumping in the upper-middle part. For scene categorization, most of the scene categories (e.g., coast, mountain, open country) have sky in the top part, which is not as discriminative as the bottom part. In scene categories of tall buildings and street, the upper part of the street consists of buildings, which discriminates these two categories from others. Not surprisingly, our method had automatically figured this out (see Figure 5b).



(a) Maps of weights ω_E given to the depth attributes for the task of event categorization



(b) Maps of weights ω_S given to the depth attributes for the task of scene categorization

Fig. 5. Figure showing the importance of depths in different regions for predicting different events/scenes. Our algorithms automatically selects the appropriate non-zero connections between the depth attributes to the final categorization tasks. An example image for each class is also shown above the map of the weights.

Now, we look at the connections between scene categorization and event categorization (Figure 6). First, it agrees with our intuitions that high weights are assigned to the outputs of the first-layer classifier that is the same as the task (e.g. The "street-like" attribute supports the recognition of a street scene). Second, the scene output supports event categorization and vice versa. This is also consistent with [14]'s intuition, where the authors (manually) decide to include the scene into the event categorization model. We also observe that in Figure 6-left, our algorithm had assigned high weights to "forest-like" attribute for the event classification tasks of "bocce", "polo" and "croquet" (these events happen in environments close to forest), and to "open-country-like" for "bocce" event, and to "coast-like" for "rowing" event, and so on. Also note that the "croquet-like" attribute actually supports the final event output of "bocce", which indicates our first-layer output learns to describe the image semantically instead of simply discriminating different classes.

Our model scales well with large number of tasks, where we want our task to depend on only a few of all the tasks (i.e. few more effective attributes). For this purpose, we used our algorithm to construct a sparse graph for connecting the tasks. For the task of event categorization, Figure 7-left shows the weight map learned for all the first-layer outputs from different classifiers —these are the input of the second-layer event classifier. We observe that all the weights corresponding to the saliency output are extremely low, which indicates that the "salient region" attribute plays little role in the final decision. Therefore we remove the link from the first-layer saliency box to the second-layer event categorization task. (We also verify that removing this link has no effect on the results.) With sparsity, the second-layer classifier input is reduced from 241-

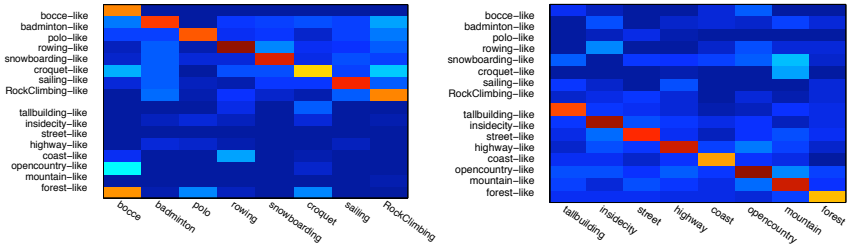


Fig. 6. (Left) Map of weights given to event and scene attributes outputs (shown in rows) for the task of event categorization (the columns), (Right) Map of weights given to them (rows) for task of scene categorization (columns). To clarify with an example, the left-bottom element in the left figure indicates the weight assigned to the first-layer score of being a forest-like scene for computing the final score of being a bocce event.

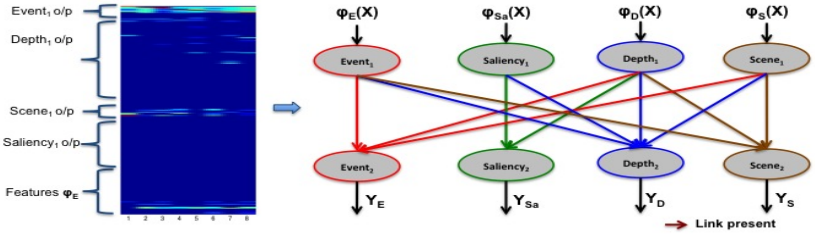


Fig. 7. (Left) The weights for the task of event categorization. (Right) The sparse graph inferred using our method.

dimension to 48-dimension. Figure 7-right shows the removed links for all other tasks overlaid on one figure. We note that depth is important for all the tasks, and predicting depth from single image also requires information from all the tasks. (This is consistent with [26] where they observed that a lot of information and expressive power is needed for the task of single image depth prediction.)

6 Conclusions

We present a generic cascaded model for connecting different vision tasks to aid holistic scene understanding. Our model considers the intermediate layer as a hidden layer, whose output can be suitably inferred through feedback from the latter layer. Prior work by Heitz et. al. [5] provided a good framework for combining classifiers but, as they point out, gets stuck as classifiers at each layer are strictly trained towards "labels". Our feedback based learning method allows the first layer to focus on more important modes, which results in learning meaningful "attributes" for optimizing the target task. In addition, we show that our model can handle heterogenous datasets and scale well to numerous vision tasks by using sparsity. We have successfully applied the *same* learning algorithm to four *different* vision tasks: saliency detection, event categorization, depth estimation and scene categorization. In our extensive experiments we show that our method improved performance in *each* of the tasks over state-of-the-art classifiers as well as previous methods of combining different vision tasks.

References

1. Saxena, A., Chung, S.H., Ng, A.Y.: 3-d depth reconstruction from a single still image. IJCV **76** (2007) 53–69

2. Li, L.J., Socher, R., Fei-Fei, L.: Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In: CVPR. (2009)
3. Hoiem, D., Efros, A.A., Hebert, M.: Closing the loop on scene interpretation. In: CVPR. (2008)
4. Sudderth, E.B., Torralba, A., Freeman, W.T., Willsky, A.S.: Depth from familiar objects: A hierarchical model for 3d scenes. In: CVPR. (2006)
5. Heitz, G., Gould, S., Saxena, A., Koller, D.: Cascaded classification models: Combining models for holistic scene understanding. In: NIPS. (2008)
6. Ferrari, V., Zisserman, A.: Learning visual attributes. In: NIPS. (2007)
7. Farhadi, A., Endres, I., Hoiem, D., Forsyth, D.: Describing objects by their attributes. In: CVPR. (2009)
8. Lampert, C., Nickisch, H., Harmeling, S.: Learning to detect unseen object classes by between-class attribute transfer. In: CVPR. (2009)
9. Wang, G., Forsyth, D.: Joint learning of visual attributes, object classes and visual saliency. In: ICCV. (2009)
10. Sudderth, E.B., Torralba, A., Freeman, W.T., Willsky, A.S.: Learning hierarchical models of scenes, objects, and parts. In: ICCV. (2005)
11. Hoiem, D., Efros, A.A., Hebert, M.: Putting objects in perspective. In: CVPR. (2006)
12. Tu, Z., Chen, X., Yuille, A.L., Zhu, S.: Image parsing: Unifying segmentation, detection, and recognition. ICCV (2003)
13. Kumar, S., Hebert, M.: A hierarchical field framework for unified context-based classification. In: ICCV. (2005)
14. Li, L.J., Fei-Fei, L.: What, where and who? classifying event by scene and object recognition. In: ICCV. (2007)
15. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: EuroCOLT. (1995)
16. Lee, H., Battle, A., Raina, R., Ng, A.Y.: Efficient sparse coding algorithms. In: NIPS. (2007)
17. Mairal, J., Leordeanu, M., Bach, F., Hebert, M., Ponce, J.: Discriminative sparse image models for class-specific edge detection and image interpretation. In: ECCV. (2008)
18. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. J of Royal Stat. Soc., Series B **39** (1977) 1–38
19. Gibbs, M., Mackay, D.: Variational gaussian process classifiers. IEEE Transactions on Neural Networks **11** (1997) 1458–1464
20. Torralba, A., Oliva, A., Castelhano, M.S., Henderson, J.M.: Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search. Psychol Rev **113** (2006) 766–786
21. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. PAMI **99** (2009)
22. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: Pascal (voc2008). http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html (2008)
23. Oliva, A., Torralba, A.: Modeling the shape of the scene: A holistic representation of the spatial envelope. IJCV **42** (2001) 145–175
24. Oliva, A., Torralba, A.: Mit outdoor scene dataset. http://people.csail.mit.edu/torralba/code/spatialenvelope/ (2009)
25. Achanta, R., Hemami, S., Estrada, F., Susstrunk, S.: Frequency-tuned Salient Region Detection. In: CVPR. (2009)
26. Saxena, A., Sun, M., Ng, A.: Make3d: Learning 3d scene structure from a single still image. PAMI **31** (2009) 824–840