Towards Holistic Scene Understanding: Feedback Enabled Cascaded Classification Models

Congcong Li, *Student Member, IEEE,* Adarsh Kowdle, *Student Member, IEEE,* Ashutosh Saxena, *Member, IEEE,* Tsuhan Chen, *Fellow, IEEE*

Abstract—Scene understanding includes many related sub-tasks, such as scene categorization, depth estimation, object detection, etc. Each of these sub-tasks is often notoriously hard, and state-of-the-art classifiers already exist for many of them. These classifiers operate on the same raw image and provide correlated outputs. It is desirable to have an algorithm that can capture such correlation without requiring any changes to the inner workings of any classifier.
We propose Feedback Enabled Cascaded Classification Models (FE-CCM), that jointly optimizes all the sub-tasks, while requiring only a 'black-box' interface to the original classifier for each sub-task. We use a two-layer cascade of classifiers, which are repeated instantiations of the original ones, with the output of the first layer fed into the second layer as input. Our training method involves a feedback step that allows later classifiers to provide earlier classifiers information about which error modes to focus on. We show that our method significantly improves performance in *all* the sub-tasks in the domain of scene understanding, where we consider depth estimation, scene categorization, event categorization, object detection, geometric labeling and saliency detection. Our method also improves performance in two robotic applications: an object-grasping robot and an object-finding robot.

Index Terms—Scene understanding, Classification, Machine learning, Robotics.

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. (See Figure 1.) Each of these tasks explains some aspect of a particular scene and in order to fully understand a scene, we would need to solve for each of these sub-tasks. Several independent efforts have resulted in good classifiers for each of these sub-tasks. In practice, we see that the sub-tasks are coupled—for example, if we know that the scene is an indoor scene, it would help us estimate depth from that single image more accurately. In another example in the robotic grasping domain, if we know what kind of object we are trying to grasp, then it is easier for a robot to figure out how to pick it up. In this paper, we propose a unified model that jointly optimizes for all the sub-tasks, allowing them to share information and guide the classifiers towards a joint optimal. We show that this can be seamlessly applied across different applications.

Recently, several approaches have tried to combine these different classifiers for related tasks in vision [1–10]; however, most of them tend to be ad-hoc (i.e., a hard-coded rule is used) and often an intimate knowledge of the inner workings of the individual classifiers is required. Even beyond vision, in many other domains, state-of-the-art classifiers already exist for many sub-tasks. However, these carefully engineered models are often tricky to modify, or even to simply re-implement from the available descriptions. Heitz et. al. [11] recently developed a framework for scene understanding called Cascaded Classification Models (**CCM**) treating each classifier as a 'black-box'. Each classifier is repeatedly instantiated with the next layer using the outputs of the previous classifiers as inputs. While this work proposed a method of combining the classifiers in a way that increased the performance in all of the four tasks they considered, it had a drawback that it optimized for each task independently and there was no way of feeding back information from later classifiers to earlier classifiers during training. This feedback can potentially help the CCM achieve a more optimal solution.

In our work, we propose Feedback Enabled Cascaded Classification Models (**FE-CCM**), which provides feedback from the later classifiers to the earlier ones, during the training phase. This feedback, provides earlier stages information about what error modes should be focused on, or what can be ignored without hurting the performance of the later classifiers. For example, misclassifying a street scene as highway may not hurt as much as misclassifying a street scene as open country. Therefore we prefer the first layer classifier to focus on fixing the latter error instead of optimizing the training accuracy. In another example, allowing the depth estimation to focus on some specific regions can help perform better scene categorization. For instance, the open country scene is characterized by its upper part as a wide sky area. Therefore, estimating the

- *Congcong Li, Adarsh Kowdle and Tsuhan Chen are with the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY, 14853. E-mail: {cl758, apk64}@cornell.edu, tsuhan@ece.cornell.edu*
- *Ashutosh Saxena is with the Department of Computer Science, Cornell University, Ithaca, NY, 14853. E-mail: asaxena@cs.cornell.edu*

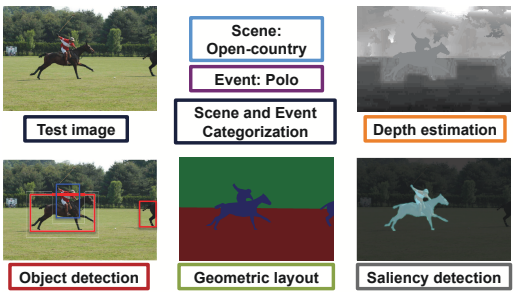

Fig. 1. Given a test image, *Holistic Scene Understanding* corresponds to inferring the labels for all possible scene understanding dimensions. In our work, we infer labels corresponding to, scene categorization, event categorization, depth estimation (Black = close, white = far), object detection, geometric layout (green = vertical, red = horizontal, blue = vertical) and saliency detection (cyan = salient) as shown above and achieve this jointly using one unified model. Note that different tasks help each other, for example, the depth estimate of the scene can help the object detector look for the horse; the object detection can help perform better saliency detection, etc.

depth well in that region by sacrificing some regions in the bottom may help to correctly classify an image. In detail, we do so by jointly optimizing all the tasks; the outputs of the first layers are treated as latent variables and training is done using an iterative algorithm. Another benefit of our method is that each of the classifiers can be trained using their own independent training datasets, i.e., our model does not require a datapoint to have labels for all the sub-tasks, and hence it scales well with *heterogeneous* datasets.

In our approach, we treat each classifier as a 'black-box', with no restrictions on its operation other than requiring the ability to train on data and have an input/output interface. (Often each of these individual classifier could be quite complex, e.g., producing labelings over pixels in an entire image.) Therefore, our method is applicable to many other tasks that have different but correlated outputs.

In extensive experiments, we show that our method achieves significant improvements in the performance of *all* the six sub-tasks we consider: depth estimation, object detection, scene categorization, event categorization, geometric labeling and saliency detection. We also successfully apply the same model to two robotics applications: robotic grasping, and robotic object detection.

The rest of the paper is organized as follows. We first define holistic scene understanding and discuss the related works in Section 2. We describe our FE-CCM method in Section 3 followed by the discussion about handling heterogeneous datasets in Section 4. We provide the implementation details of the classifiers in Section 5. We present the experiments and results in Section 6 and some robotic applications in Section 7. We finally conclude in Section 8.

2 Overview of Scene Understanding

2.1 Holistic Scene Understanding

When we look at an image of a scene, such as in Figure 1, we are often interested in answering several different questions: What objects are there in the image? How far are things? What is going on in the scene? What type of scene it is? And so on. These are only a few examples of questions in the area of scene understanding; and there may even be more.

In the past, the focus has been to address each task in isolation, where the goal of each task is to produce a label $Y_i \in \mathbf{S_i}$ for the i^{th} sub-task. If we are considering depth estimation (see Figure 1), then the label would be $Y_1 \in \mathbf{S_1} = \mathbb{R}_+^{100\times100}$ for continuous values of depth in a 100×100 output. For scene categorization, we will have $Y_2 \in \mathbf{S_2} = \{1, \ldots, K\}$ for K scene classes. If we have n sub-tasks, then we would have to produce an output as:

$$\mathcal{Y} = \{Y_1, \ldots, Y_n\} \in \mathbf{S_1} \times \mathbf{S_2} \ldots \times \mathbf{S_n}.$$

The interesting part here is that often we want to solve different combinations of the sub-tasks depending on the situation. The goal of this work is to design an algorithm that does not depend on the particular sub-tasks in question.

2.2 Related Work

Cascaded classifiers. Using information from related tasks to improve the performance of the task in question has been studied in various fields of machine learning. The idea of cascading layers of classifiers to aid a task was first introduced with neural networks as multi-level perceptrons where, the output of the first layer of perceptrons is passed on as input to the next layer [12–14]. However, it is often hard to train neural networks and gain an insight into its operation, making it hard to work for complicated tasks.

The idea of improving classification performance by combining outputs of many classifiers is used in methods such as Boosting [15], where many weak learners are combined to obtain a more accurate classifier; this has been applied to tasks such as face detection [16, 17]. To incorporate contextual information, Fink and Perona [18] exploited local dependencies between objects in a boosting framework, but did not allow for multiple rounds of communication between objects. Torralba et al. [19] introduced Boosted Random Fields to model object dependency, which used boosting to learn the graph structure and local evidence of a conditional random field. Tu [20] proposed a more general framework which used pixel-level label maps to learn a contextual model through a cascaded classifier approach. All these works mainly consider the interactions between labels of the same type. However, in our CCM framework [21, 22], the focus is on capturing contextual interactions between labels of different types. Furthermore, compared to the feed-forward only cascade method in [20], our model with feedback not only iteratively refines the contextual interactions, but also refines the individual classifiers to provide helpful context.

Sensor fusion. There has been a huge body of work in the area of sensor fusion where classifiers output the same labels but work with different modalities, each one giving additional information and thus improving the performance, e.g., in biometrics, data from voice recognition and face recognition is combined [23]. However, in our scenario, we consider multiple tasks where each classifier is tackling a different problem (i.e., predicting different labels), with the same input being provided to all the classifiers.

Structured Models for combining tasks. While the methods discussed above combine classifiers to predict the *same* labels, there is a group of works that designs models

for predicting heterogenous labels. Kumar and Hebert [1] developed a large MRF-based probabilistic model to link multi-class segmentation and object detection. Li et al. [24] modeled mulitple interactions within tasks and across tasks by defining a MRF over parameters. Similar efforts have been made in the field of natural language processing. Sutton and McCallum [6] combined a parsing model with a semantic role labeling model into a unified probabilistic framework that solved both simultaneously. Ando and Zhang [25] proposed a general framework for learning predictive functional structures from multiple tasks. All these models require knowledge of the inner workings of the individual classifiers, which makes it hard to fit existing state-of-the-art classifiers of certain tasks into the models.

Structured learning algorithms (e.g., [26–28]) can also be a viable option for the setting of combining multiple tasks. There has been recent development in structured learning on handling latent variables (e.g. hidden conditional random field [29], latent structured SVM [30]), which can be potentially applied to multi-task settings with disjoint datasets. With considerable understanding into each of the tasks, the loss function in structured learning provides a nice way to leverage different tasks. However, in this work, we focus on developing a more generic algorithm that can be easily applied even without intimate knowledge of the tasks.

There have been many works which show that with a well-designed model, one can improve the performance of a particular task by using cues from other tasks (e.g., [7–9]). Saxena et al. manually designed the terms in an MRF to combine depth estimation with object detection [2] and stereo cues [10]. Sudderth et al. [5] used object recognition to help 3D structure estimation.

Context. There is a large body of work that leverages contextual information to help specific tasks. Various sources of context have been explored, ranging from the global scene layout, interactions between objects and regions to local features. To incorporate scene-level information, Torralba et al. [31, 32] used the statistics of low-level features across the entire scene to prime object detection or help depth estimation. Hoiem et al. [33] used 3D scene information to provide priors on potential object locations. Park et al. [34] used the ground plane estimation as contextual information for pedestrian detection. Many works also model context to capture the local interactions between neighboring regions [35–37], objects [38–42], or both [43–45]. These methods improve the performance of some specific tasks by combining information from different aspects. However, most of these methods can not be applied to cases when we only have "black-box" classifiers for the individual tasks.

Holistic Scene Understanding. Hoiem et. al. [3] proposed an innovative but ad-hoc system that combined boundary detection and surface labeling by sharing some low-level information between the classifiers. Li et. al. [4, 46] combined image classification, annotation and segmentation with a hierarchical graphical model. However, these methods required considerable attention to each classifier, and considerable insight into the inner workings of each task and also the connections between them. This limits the generality of the approaches in introducing new tasks easily or being applied to other domains.

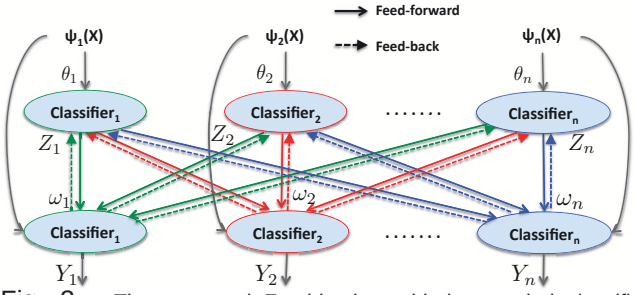

Fig. 2. The proposed Feed-back enabled cascaded classification model (FE-CCM) for combining related classifiers. ($\forall i \in \{1, 2, \ldots, n\}$, $\Psi_i(X)$ = Features corresponding to $Classifier_i$ extracted from image X, Z_i = Output of the $Classifier_i$ in the first stage parameterized by θ_i, Y_i = Output of the $Classifier_i$ in the second stage parameterized by ω_i). In the proposed FE-CCM model, there is feed-back from the latter stages to help achieve a model which optimizes all the tasks considered, jointly. Here $Classifier_i$'s on the two layers can have different forms though they are for the same classification task. (Note that different colors of lines are used only to make the figure more readable.)

Deep Learning. There is also a large body of work in the areas of deep learning, and we refer the reader to Bengio and LeCun [47] for a nice overview of deep learning architectures and Caruana [48] for multitask learning with shared representation. While efficient back-propagation methods like [49] have been commonly used in learning a multi-layer network, it is not as easy to apply to our case where each node is a complex classifier. Most works in deep learning (e.g., [50–52]) are different from our work in that, those works focus on one particular task (same labels) by building different classifier architectures, as compared to our setting of *different* tasks with different labels. Hinton et al. [51] used unsupervised learning to obtain an initial configuration of the parameters. This provides a good initialization and hence their multi-layered architecture does not suffer from local minimas during optimization. At a high-level, we can also look at our work as a multi-layered architecture (where each node typically produces complex outputs, e.g., labels over the pixels in the image); and initialization in our case comes from existing state-of-the-art individual classifiers. Given this initialization, our training procedure finds parameters that (consistently) improve performance across all the sub-tasks.

3 Feedback Enabled Cascaded Classification Models

In the field of scene understanding, a lot of independent research into each of the vision sub-tasks has led to excellent classifiers. These independent classifiers are typically trained on different or *heterogenous* datasets due to the lack of ground-truth labels for all the sub-tasks. In addition, each of these classifiers come with their own learning and inference methods. Our goal is to consider each of them as a 'black-box', which makes it easy to combine them. We describe what we mean by 'black-box classifiers' below.

Black-box Classifier. A black-box classifier, as the name suggests, is a classifier for which operations (such as learning and inference algorithms) are available for use, but their inner workings are not known. We assume that,

given the training dataset $\mathbf{X}$, features extracted $\mathbf{\Psi}(\mathbf{X})$ and the target outputs of the i^{th} task $\mathbf{Y_i}$, the black-box classifier has some internal learning function f^i_{learn} with parameters θ_i that optimizes the mapping from the inputs to the outputs for the training data. [1] Once the parameters have been learnt, given a new data point, X with features $\Psi(X) \in \mathbb{R}^K$, where K can be changed as desired,[2] the black-box classifier returns the output $\hat{Y_i}$ according to its internal inference function f^i_{infer}. This is illustrated through the equations below. For the i^{th} task,

$$\textit{Learning} : \theta_i^* = \underset{\theta_i}{\text{optimize}}\, f^i_{\text{learn}}(\mathbf{\Psi}(\mathbf{X}), \mathbf{Y_i}; \theta_i) \quad (1)$$

$$\textit{Inference} : \hat{Y_i} = \underset{Y_i}{\text{optimize}}\, f^i_{\text{infer}}(\Psi(X), Y_i; \theta_i^*). \quad (2)$$

This approach of treating each classifier as a black-box allows us to use different existing classifiers which have been known to perform well at specific sub-tasks. Furthermore, without changing their inner workings, it allows us to compose them into one model which exploits the information from each sub-task to aid holistic scene understanding.

3.1 Our Model

Our model is built in the form of a two-layer cascade, as shown in Figure 2. The first layer consists of an instantiation of each of the black-box classifiers with the image features as input. The second layer is a repeated instantiation of each of the classifiers with the first layer classifier outputs as well as the image features as inputs. Note that the repeated classifier on the second layer is not necessary to have the same mathematical form with the one on the first layer. Instead, we consider it as a repeated instantiation only because they are used for the same classification task.

Notation: We consider n related sub-tasks Classifier_i, $i \in \{1, 2, \ldots, n\}$ (Figure 2). We describe the notations used in this paper as follows:

$\Psi_i(X)$	Features corresponding to Classifier_i extracted from image X.
Z_i, $\mathcal{Z}$	Z_i indicates output from the first layer Classifier_i. Many classifiers output continuous scores instead of labels. In cases where this is not the case, it is trivial to convert a binary classiers output to a log-odds scores. For a K-class ($K > 2$) classifier, we consider the output to be a K-dimensional vector. $\mathcal{Z}$ indicates the set $\{Z_1, Z_2, \ldots, Z_n\}$.
θ_i, Θ	θ_i indicates parameters corresponding to first layer Classifier_i. Θ indicates the set $\{\theta_1, \ldots, \theta_n\}$.
Y_j, $\mathcal{Y}$	Y_j indicates output for the j^{th} task in the second layer, using the original features $\Psi_j(X)$ as well as all the outputs from the first layer as input. $\mathcal{Y}$ indicates the set $\{Y_1, Y_2, \ldots, Y_n\}$.
ω_j, Ω	ω_j indicates parameters for the second layer Classifier_j. Ω indicates the set $\{\omega_1, \ldots, \omega_n\}$.
Γ_j, Γ	Dataset for the j^{th} task, which consists of labeled pairs $\{X, Y_j\}$ in the training set. Γ represents all the labeled data.
f^i_{infer}, f^i_{learn}	the internal inference function and learning function for the i^{th} classifier on the first layer.
f'^i_{infer}, f'^i_{learn}	the internal inference function and learning function for the i^{th} classifier on the second layer.

1. Unless specified, the regular symbols (e.g. X, Y_i, etc.) are used for a particular data-point, and the bold-face symbols (e.g. $\mathbf{X}$, $\mathbf{Y_i}$, etc.) are used for a dataset.

2. If the input dimension of the black-box classifier can not be changed, then we will use that black-box in the first layer only.

With the notations in place we will now first describe the inference and learning algorithms for the proposed model in the following sections, followed by probabilistic interpretation of our method.

3.2 Inference Algorithm

During inference, the inputs $\Psi_i(X)$ are given and our goal is to infer the final outputs Y_i. Using the learned parameters θ_i for the first level of classifiers and ω_i for the second level of classifiers, we first infer the first-layer outputs Z_i and then infer the second-layer outputs Y_i. More formally, we perform the following.

$$\hat{Z_i} = \underset{Z_i}{\text{optimize}}\, f^i_{\text{infer}}(\Psi_i(X), Z_i; \hat{\theta}_i) \quad (3)$$

$$\hat{Y_i} = \underset{Y_i}{\text{optimize}}\, f'^i_{\text{infer}}([\Psi_i(X)\ \hat{\mathcal{Z}}], Y_i; \hat{\omega}_i) \quad (4)$$

The inference algorithm is given in Algorithm 1. This method allows us to use the internal inference function (Equation 2) of the black-box classifiers without knowing its inner workings. Note that the complexity here is no more than constant times the complexity of inference in the original classiers.

Algorithm 1 Inference

1. **Inference for first layer:**
   for $i = 1 : n$
   Infer the outputs of the i^{th} classifier using Equation 3;
   end
2. **Inference for second layer:**
   for $i = 1 : n$
   Infer the outputs of the i^{th} classifier using Equation 4;
   end

3.3 Learning Algorithm

During the training stage, the inputs $\Psi_i(X)$ as well as the target outputs, $Y_1, Y_2, \ldots, Y_n$ of the second level of classifiers, are all observed (because the ground-truth labels are available). In our algorithm, we consider $\mathcal{Z}$ (outputs of layer 1 and inputs to layer 2) as hidden variables. In previous work, Heitz et al. [11] assume that each layer is independent and that each layer produces the best output independently (without consideration for other layers), and therefore use the ground-truth labels for $\mathcal{Z}$ even for training the classifiers in the first layer.

On the other hand, we want to optimize for the final outputs as much as possible. Thus the first layer classifiers need not perform their best (w.r.t. groundtruth), but rather focus on error modes that would result in the second layer's output $(Y_1, Y_2, \ldots, Y_n)$ being more correct. Therefore, we learn the model through an iterative Expectation-Maximization formulation, given the independencies between classifiers represented by the model in Figure 2. In one step (Feed-forward step) we assume the variables Z_i's are known and learn the parameters and in the other step (Feed-back step) we fix the parameters estimated previously and estimate the variables Z_i's. Since the Z_i's are not fixed to the ground truth, as the iterations progress, the first level of classifiers start focusing on the error modes which would give the best improvement in performance at the end of the second level of classifiers. The learning algorithm is summarized in Algorithm 2.

Algorithm 2 Learning

1. Initialize latent variables $\mathcal{Z}$ with the ground-truth $\mathcal{Y}$.

2. Do until convergence or maximum iteration: {

Feed-foward step: Fix latent variables $\mathcal{Z}$, estimate the parameters Θ and Ω using Equation 5 and Equation 6.

Feed-back step: Fix the parameters Θ and Ω, compute latent variables $\mathcal{Z}$ using Equation 7.

}

Initialization: We initialize the model by setting the latent variables Z_i's to the groundtruth, i.e. $Z_i = Z_i^{gt}$. Training with this initialization, our cascade is equivalent to CCM in [11], where the classifiers (and the parameters) in the first layer are similar to the original state-of-the-art classifier and the classifiers in the second layer use the outputs of the first layer in addition to the original features as input.

Feed-forward Step: In this step, we estimate the parameters Θ and Ω. We assume that the latent variables Z_i's are known (and Y_i's are known because they are the ground-truth during learning, i.e. $Y_i = Y_i^{gt}$). We then learn the parameters of each classifier independently. Learning θ_i is precisely the learning problem of the 'black-box classifier', and learning ω_i is also an instantiation of the original learning problem, but with the original input features appended with the outputs of the first level classifiers. Therefore, we can use the learning method provided by the individual black-box classifier (Equation 1).

$$\hat{\theta}_i = \underset{\theta_i}{\text{optimize}}\, f^i_{\text{learn}}(\mathbf{\Psi_i}(\mathbf{X}), \mathbf{Z}_i; \theta_i) \tag{5}$$

$$\hat{\omega}_i = \underset{\omega_i}{\text{optimize}}\, f'^i_{\text{learn}}([\mathbf{\Psi_i}(\mathbf{X})\ \boldsymbol{\mathcal{Z}}], \mathbf{Y}_i; \omega_i) \tag{6}$$

We now have the parameters for all the classifiers.

Feed-back Step: In the second step, we will estimate the values of the variables Z_i's assuming that the parameters are fixed (and Y_i's are given because the ground-truth is available, i.e. $Y_i = Y_i^{gt}$). This feed-back step is the crux that provides information to the first-layer classifiers what error modes should be focused on and what can be ignored without hurting the final performance. Given θ_i's and ω_i's are fixed, we want the Z_i's to be good predictions from the first-layer classifiers and also help to increase the correction predictions of Y_i's as much as possible. We optimize the following function for the feed-back step:

$$\underset{\mathcal{Z}}{\text{optimize}} \sum_{i=1}^{n} \Big(J^i_1(\Psi_i(X), Z_i; \hat{\theta}_i) + J^i_2(\Psi_i(X), \mathcal{Z}, Y_i; \hat{\omega}_i) \Big) \tag{7}$$

where J^i_1's and J^i_2's are functions respectively related to the first-layer classifiers and the second-layer classifiers. one option is to have $J^i_1(\Psi_i(X), Z_i; \hat{\theta}_i) = f^i_{\text{infer}}(\Psi_i(X), Z_i; \hat{\theta}_i)$ and $J^i_2(\Psi_i(X), \mathcal{Z}, Y_i; \hat{\omega}_i) = f'^i_{\text{infer}}([\Psi_i(X), \mathcal{Z}], Y_i; \hat{\omega}_i)$ if the intrinsic inference functions for the classifiers are known. More discussions will be given in Section 3.4 if the intrinsic functions are unknown. The updated Z_i's will be used to re-learn the classifier parameters in the feed-forward step of next iteration. Note that the updated Z_i's have continuous values. If the internal learning function of a classifier accepts only labels, we threshold the values of Z_i's to get labels.

3.4 Probabilistic Interpretation

Our algorithm can be explained with a probabilistic interpretation where the goal is to maximize the log-likelihood of the outputs of all tasks given the observed inputs, i.e., $\log P(\mathcal{Y}|X)$, where X is an image belonging to training set Γ. Therefore, the goal of the proposed model shown in Figure 2 is to maximize

$$\log \prod_{X \in \Gamma} P(\mathcal{Y}|X; \Theta, \Omega) \tag{8}$$

To introduce the hidden valuables Z_i's, we expand Equation 8 as follows, using the independencies represented by the directed model in Figure 2.

$$= \sum_{X \in \Gamma} \log \sum_{\mathcal{Z}} P(Y_1, \ldots, Y_n, \mathcal{Z}|X; \Theta, \Omega) \tag{9}$$

$$= \sum_{X \in \Gamma} \log \sum_{\mathcal{Z}} \prod_{i=1}^{n} P(Y_i|\Psi_i(X), \mathcal{Z}; \omega_i) P(Z_i|\Psi_i(X); \theta_i) \tag{10}$$

However, the summation inside the $\log$ makes it difficult to learn the parameters. Motivated by the Expectation Maximization algorithm [53], we iterate between the two steps as described in the following. Again we initialize the classifiers by learning the classifiers with ground-truth as discussed Section 3.3.

Feed-forward Step: In this step, we estimate the parameters by assuming that the latent variables Z_i's are known (and Y_i's are known anyway because they are the ground-truth). This results in

$$\underset{\theta_1, \ldots, \theta_n, \omega_1, \ldots, \omega_n}{\text{maximize}} \sum_{X \in \Gamma} \log \prod_{i=1}^{n} P(Y_i|\Psi_i(X), \mathcal{Z}; \omega_i) P(Z_i|\Psi_i(X); \theta_i) \tag{11}$$

Now in this feed-forward step, the terms for maximizing the different parameters turn out to be independent. So, for the i^{th} classifier we have:

$$\underset{\theta_i}{\text{maximize}} \sum_{X \in \Gamma} \log P(Z_i|\Psi_i(X); \theta_i) \tag{12}$$

$$\underset{\omega_i}{\text{maximize}} \sum_{X \in \Gamma} \log P(Y_i|\Psi_i(X), \mathcal{Z}; \omega_i) \tag{13}$$

Note that the optimization problem nicely breaks down into the sub-problems of training the individual classifier for the respective sub-tasks. We can solve each sub-problem separately given the probabilistic interpretation of the corresponding classifier. When the classifier is taken as 'black-box', this can be approximated using the original learning method provided by the individual black-box classifier (Equation 5 and Equation 6)

Feed-back Step: In this step, we estimate the values of the latent variables Z_i's assuming that the parameters are fixed. We perform MAP inference on Z_i's (and not marginalization). This can be considered as a special variant of the general EM framework (hard EM, [54]). Using Equation 10, we get the following optimization problem:

$$\underset{\mathcal{Z}}{\text{maximize}} \log P(Y_1, \ldots, Y_n, \mathcal{Z}|X; \hat{\theta}_1, \ldots, \hat{\theta}_n, \hat{\omega}_1, \ldots, \hat{\omega}_n) \Leftrightarrow$$

$$\underset{\mathcal{Z}}{\text{maximize}} \sum_{i=1}^{n} \Big(\log P(Y_i|\Psi_i(X), \mathcal{Z}; \hat{\omega}_i) + \log P(Z_i|\Psi_i(X); \hat{\theta}_i) \Big) \tag{14}$$

This maximization problem requires that we have access to the characterization of the individual black-box classifiers

in a probabilistic form. If the probabilistic interpretations of the classifiers are known, we can solve the above function accordingly. Note that Equation 14 is same as Equation 7 with $J_1^i(\Psi_i(X), Z_i; \hat{\theta}_i) = \log P(Z_i|\Psi_i(X); \hat{\theta}_i)$ and $J_2^i(\Psi_i(X), \mathcal{Z}, Y_i; \hat{\omega}_i) = \log P(Y_i|\Psi_i(X), \mathcal{Z}; \hat{\omega}_i)$.

In some cases, the classifier log-likelihoods in Equation 14 actually turn out to be convex. For example, if the individual classifiers are linear or logistic classifiers, the minimization problem is convex and can be solved using gradient descent (or any such method).

However, if the probabilistic interpretations of the classifiers are unknown, the feedback step requires extra modeling. Some modeling options are provided as follows:

- Case 1: Insight into the vision problem is available. In this case, one could use the domain knowledge of the task into the problem to properly model J_1^i's and J_2^i's.
- Case 2: No insight into the vision problem is available and no internal function of the original classifier is known. In this case, we formulate the J_1^i's and J_2^i's as follows. The J_1^i is defined to be a distance function between the target Z_i and the estimated $\hat{Z}_i$, which serves as a regularization for the first-layer classifiers.

$$\begin{aligned} J_1^i(\Psi_i(X), Z_i; \hat{\theta}_i) &= \left|\left|Z_i - \hat{Z}_i\right|\right|^2 \\ \text{s.t. } \hat{Z}_i &= \underset{Z_i}{\text{optimize}}\, f^i_{\text{infer}}(\Psi_i(X), Z_i; \hat{\theta}_i) \end{aligned} \tag{15}$$

To formulate J_2^i's, we make a variational approximation on the output of the second-layer classifier for task i (i.e., approximating it as a Gaussian, [55]) to get:

$$\underset{\alpha_i}{\text{minimize}} \sum_{X \in \Gamma} \left|\left|\hat{Y}_i - \alpha_i^T[\Psi_i(X), \hat{\mathcal{Z}}]\right|\right|_2^2 \tag{16}$$

where α_i are parameters of the approximation model. $\hat{Y}_i$ is the actual output of the second layer classifier for the task i, i.e. $\hat{Y}_i = \text{optimize}_{Y_i} f'_{\text{infer}}([\Psi_i(X)\ \ \hat{\mathcal{Z}}], Y_i; \hat{\omega}_i)$. Then we define the J_2^i's as follows.

$$J_2^i(\Psi_i(X), \mathcal{Z}, Y_i; \hat{\omega}_i) = \left|\left|Y_i - \hat{\alpha}_i^T[\Psi_i(X), \mathcal{Z}]\right|\right|_2^2 \tag{17}$$

Sparsity: Note that the parameter α_i 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 for the approximation model may become ill-conditioned. Therefore, we want our model to select only a few non-zero weights, i.e., only a few non-zero entries in α_i. We do this by introducing the l_1 sparsity in the parameters [56]. So Equation 16 is extended as follows.

$$\underset{\alpha_i}{\text{minimize}} \sum_{X \in \Gamma} \left(\left|\left|\hat{Y}_i - \alpha_i^T[\Psi_i(X), \hat{\mathcal{Z}}]\right|\right|_2^2 + \beta \left|\alpha_i\right| \right) \tag{18}$$

Inference: As introduced in Section 3.2, our inference procedure consists of two steps: first maximize over hidden variable Z and then maximize over Y. [3]

$$\hat{\mathcal{Z}} = \underset{\mathcal{Z}}{\text{argmax}} \log P(\mathcal{Z}|X, \hat{\Theta}) \tag{19}$$

$$\hat{\mathcal{Y}} = \underset{\mathcal{Y}}{\text{argmax}} \log P(\mathcal{Y}|\hat{\mathcal{Z}}, X, \hat{\Omega}) \tag{20}$$

3. Another alternative would have been to maximize $P(Y|X) = \sum_Z P(Y, Z|X)$; however, this would require marginalization over the variable Z which is expensive to compute.

Given the structure of our directed graph, the outputs for different classifiers on the same layer are independent given their inputs and parameters. Therefore, Equations 19 and 20 are equivalent to the following:

$$\hat{Z}_i = \underset{Z_i}{\text{argmax}} \log P(Z_i|\Psi_i(X); \hat{\theta}_i), i = 1, \ldots, n \tag{21}$$

$$\hat{Y}_i = \underset{Y_i}{\text{argmax}} \log P(Y_i|\Psi_i(X); \hat{\mathcal{Z}}; \hat{\omega}_i), i = 1, \ldots, n \tag{22}$$

As we see, Equation 21 and Equation 22 are instantiations of Equation 3 and Equation 4 in the probabilistic form.

4 Training with Heterogeneous Datasets

Often real datasets are disjoint for different tasks, i.e, each datapoint does not have the labels for all the tasks. Our formulation handles this scenario well. In this section, we show our formulation for this general case, where we use Γ_i as the dataset that has labels only for the i^{th} task.

In the following we provide the modifications to the feed-forward step and the feed-back step while dealing with disjoint datasets, i.e., data in dataset Γ_i only have labels for the i^{th} task. These modifications also allow us to develop different variants of the model, described in Section 4.1.

Feed-forward Step: Using the feedback step, we can have Z_i's for all the data. Therefore, we use all the datasets in order to re-learn each of the first-layer classifiers. If the internal learning function of the black-box classifier is additive over the data points, then we have

$$\hat{\theta}_i = \underset{\theta_i}{\text{optimize}} \sum_j \sum_{X \in \Gamma_j} \pi_j f^i_{\text{learn}}(\Psi_i(X), Z_i; \theta_i), \tag{23}$$

where π_j's are the importance factors given to different datasets, and satisfy $\sum_j \pi_j = 1$. (See Section 4.1 on how to choose π_j's.)

If the internal learning function is not additive over the data points, we provide an alternative solution here. We sample a subset of data $\mathbf{X}^j$ from each dataset Γ^j, i.e. $\mathbf{X}^j \subseteq \Gamma^j$ and combine them into a new set $\mathbf{X} = [\mathbf{X}^1, \ldots, \mathbf{X}^n]$. In $\mathbf{X}$, the ratio of data belonging to $\mathbf{X}^j$ is equal to π_j, i.e. $\frac{|\mathbf{X}^j|}{|\mathbf{X}|} = \pi_j$, where $|\cdot|$ indicates the number of data-points. Then we can learn the parameters of the first-layer classifiers as follows.

$$\hat{\theta}_i = \underset{\theta_i}{\text{optimize}}\, f^i_{\text{learn}}(\mathbf{\Psi_i}(\mathbf{X}), \mathbf{Z_i}; \theta_i), \tag{24}$$

To re-learn the second-layer classifiers, the only change made to Equation 6 is that instead of using all data while optimizing for a particular task, we use only the data-points that have the ground-truth label for the corresponding task.

$$\hat{\omega}_i = \underset{\omega_i}{\text{optimize}}\, f'^i_{\text{learn}}([\mathbf{\Psi_i}(\mathbf{X})\ \boldsymbol{\mathcal{Z}}], \mathbf{Y}_i; \omega_i), \text{ s.t. } \mathbf{X} = \Gamma_i \tag{25}$$

Feed-back Step: In this step, we change Equation 7 as follows. Since a datapoint in the set Γ_j only has ground-truth label for the j^{th} task (Y_j), we only consider J_2^j in the second term. However, since this datapoint has outputs for all the first-layer classifiers using the feed-forward step, we consider all the J_1^i's, $i = 1, \cdots, n$. Therefore, in order to obtain the value of $\mathcal{Z}$ corresponding to each data-point $X \in \Gamma_j$, we have

$$\underset{\mathcal{Z}}{\text{optimize}} \sum_{i=1}^{n} \left(J_1^i(\Psi_j(X), Z_i; \hat{\theta}_i)\right) + J_2^j(\Psi_j(X), \mathcal{Z}, Y_j; \hat{\omega}_j). \tag{26}$$

4.1 FECCM: Different Instantiations

The parameters π_j allow us to formulate three different instantiations of our model.

- **Unified FECCM**: In this instantiation, our goal is to achieve improvements in all tasks with one set of parameters $\{\Theta, \Omega\}$. We want to balance the data from different datasets (i.e., with different task labels). Towards this goal, π_j is set to be inversely proportional to the amount of data in the dataset of the j^{th} task. Therefore, the unified FECCM balances the amount of data in different datasets, based on Equation 23.

- **One-goal FECCM**: In this instantiation, we set $\pi_j = 1$ if $j = k$, and $\pi_j = 0$ otherwise. This is an extreme setting to favor the specific task k. In this case, the retraining of the first-layer classifiers will only use the feedback from the Classifier_k on the second layer, i.e., only use the dataset with labels for the k^{th} task. Therefore, FECCM degrades to a model with only one target task (the k^{th} task) on the second layer and all the other tasks are only instantiated on the first layer. Although the goal in this setting is to completely benefit the k^{th} task, in practice it often results in overfitting and does not always achieve the best results even for the specific task (see Table 1 in Section 6). In this case, we train different models, i.e. different θ_i's and ω_i's, for different target tasks.

- **Target-Specific FECCM**: This instantiation is to optimize the performance of a specific task. As compared to one-goal FECCM where we manually remove the other tasks on the second layer, in this instantiation we keep all the tasks on the second layer and conduct data-driven selection of the parameters π_j for different datasets. In detail, π_j is selected through cross validation on a hold-out set in the learning process in order to optimize the second-layer output of a specific task. Since Target-Specific FECCM still has all the tasks instantiated on the second layer, the re-training of the first-layer classifiers can still use data from different datasets (i.e., with different task labels). Here we train different models, i.e. different θ_i's and ω_i's, for different target tasks.

5 Scene Understanding: Implementation

In this section we describe the implementation details of our instantiation of **FE-CCM** for scene understanding. Each of the classifiers described below for the sub-tasks are our "base-model" shown in Table 1. In some sub-tasks, our base-model will be simpler than the state-of-the-art models (that are often hand-tuned for the specific sub-tasks respectively). However, even when using base-models in our **FE-CCM**, our model will still outperform the state-of-the-art models for the respective sub-tasks (on the same standard respective datasets) in Section 6.

In order to explain the implementation details for the different tasks, we will use the following notation. Let i be the index of the tasks we consider. We consider 6 tasks for our experiments on scene understanding: scene categorization ($i = 1$), depth estimation ($i = 2$), event categorization ($i = 3$), saliency detection ($i = 4$), object detection ($i = 5$) and geometric labeling ($i = 6$). The inputs for the j^{th} task at the first layer are given by the low-level features Ψ_j. At the second layer, in addition to the original features Ψ_j, the inputs include the outputs from the first layer classifiers. This is given by,

$$\Phi_j = [\Psi_j \; Z_1 \; Z_2 \; Z_3 \; Z_4 \; Z_5 \; Z_6] \tag{27}$$

where, Φ_j is the input feature vector for the j^{th} task on the second layer, and Z_i ($i = 1, \ldots, 6$) represents the output from the i^{th} task which is appended to the input to the j^{th} task on the second layer and so on.

Scene Categorization. For scene categorization, we classify an image into one of the 8 categories defined by Torralba et al. [57] tall building, inside city, street, highway, coast, open country, mountain and forest. We evaluate the performance by measuring the rate of incorrectly assigning a scene label to an image on the MIT outdoor scene dataset [57]. The feature inputs for the first-layer scene classifier $\Psi_1 \in \mathbb{R}^{512}$ is the GIST feature [57], extracted at 4×4 regions of the image, on 4 scales and 8 orientations.

We use an RBF-Kernel SVM classifier [58], as the first-layer scene classifier, and a multi-class logistic classifier for the second layer. The output of the first-layer scene classifier $Z_1 \in \mathbb{R}^8$ is an 8-dimensional vector where each element represents the log-odds score of the corresponding image belonging to a scene category. This 8-dimensional output is fed to each of the second-layer classifiers.

Depth Estimation. For the single image depth estimation task, we estimate the depth of every pixel in an image. We evaluate the estimation performance by computing the root mean square error of the estimated depth with respect to ground truth laser scan depth using the Make3D Range Image dataset [59, 60]. We uniformly divide each image into 55×305 patches as [59]. The feature inputs for the first-layer depth estimation $\Psi_2 \in \mathbb{R}^{104}$ are features which capture texture, color and gradient properties of the patch. This is obtained by convolving the image with Laws' masks and computing the energy and Kurtosis over the patch along with the shape features as described by Saxena et al. [59].

We use a linear regression for the first-level and second-level instantiation of the depth estimation module. The output of the first-layer depth estimation $Z_2 \in \mathbb{R}_+$ is the predicted depth of each patch in the image. In order to feed the first-layer depth output to the second-layer classifiers, for the scene categorization and event categorization tasks, we use a vector with the predicted depth of all patches in the image; for the other tasks, we use the 1-dimensional predicted depth for the patch/pixel/bounding-box, etc.

Event Categorization: For event categorization, we classify an image into one of the 8 sports events as defined by Li et al. [46]: bocce, badminton, polo, rowing, snowboarding, croquet, sailing and rock-climbing. For evaluation, we compute the rate of incorrectly assigning an event label to an image. The feature inputs for the first-layer event classifier $\Psi_3 \in \mathbb{R}^{43}$ is a 43-dimensional feature vector, which includes the top 30 PCA projections of the 512-dimensional GIST features [61], the 12-dimension global

color features (mean and variance of RGB and YCrCb color channels over the entire image), and a bias term.

We use a multi-class logistic classifier on each layer for event classification. The output of the first-layer event classifier $Z_3 \in \mathbb{R}^8$ is an 8-dimensional vector where each element represents the log-odd score of the corresponding image belonging to a event category. This 8-dimensional output is feed to each of the second-layer classifiers.

Saliency Detection. The goal of the saliency detection task is to classify each pixel in the image as either salient or non-salient. We use the saliency detection dataset used by Achanta et. al. [62] for our experiments. The feature inputs for the first-layer saliency classifier $\Psi_4 \in \mathbb{R}^4$ includes the 3-dimensional color-offset features based on the Lab color space as described by Achanta et al. [62] and a bias term.

We use a logistic model for the saliency estimation classifiers on both layers. The output of the first-layer saliency classifier $Z_4 \in \mathbb{R}$ is the log-odd score of a pixel being salient. In order to feed the first-layer saliency detection output to the second-layer classifiers, for the scene categorization and event categorization tasks, we form a vector with the predicted saliency of all the pixels in the image; for the other tasks, we use the 1-dimensional average saliency for the corresponding pixel/patch/bounding-box.

Object Detection. We consider the following object categories: car, person, horse and cow. We use the train-set and test-set of PASCAL 2006 [63] for our experiments. Our object detection module builds on the part-based detector of Felzenszwalb et. al. [64]. We first generate 5 to 100 candidate windows for each image by applying the part-based detector with a low threshold (over-detection). The feature inputs for the first-layer object detection classifier $\Psi_5 \in \mathbb{R}^K$ are the HOG features extracted based on the candidate window as [65] plus the detection score from the part-based detector [64]. K depends on the number of scales to be considered and the size of the object template.

We learn an RBF-kernel SVM model as the first layer classifier. The classifier assigns each window a $+1$ or 0 label indicating whether the window belongs to the object or not. For the second-layer classifier, we learn a logistic model over the feature vector constituted by the outputs of all first-level tasks and the original HOG feature. We use average precision to quantitatively measure the performance. The output of the first-layer object detection classifier $Z_5 \in \mathbb{R}^4$ are the estimated 0 or 1 labels for a region to belong to the 4 object categories we consider. In order to feed the first-layer object detection output to the second-layer classifiers, we first generate a detection map for each object. Pixels inside the estimated positive boxes are labeled as "+1", otherwise they are labeled as "0". For scene categorization and event categorization on the second layer, we feed all the elements on the map; for the other tasks, we use the 1-dimensional average value on the map for the corresponding pixel/patch/bounding-box.

Geometric labeling. The geometric labeling task refers to assigning each pixel to one of three geometric classes: support, vertical and sky, as defined by Hoiem et al. [33]. For evaluation, we compute the accuracy of assigning the correct geometric label to a pixel. The feature inputs for the first-layer geometry labeling classifier $\Psi_6 \in \mathbb{R}^{52}$ are the region-based features as described by Hoiem et al. [33].

We use the dataset and the algorithm by [33] as the first-layer geometric labeling module. To reduce the computation time, we avoid the multiple segmentations and instead use a single segmentation with 100 segments per image. We use a logistic model as the second-layer classifier. The output of the first-layer geometry classifier $Z_6 \in \mathbb{R}^3$ is a 3-dimensional vector with each element representing the log-odd score of the corresponding pixel belonging to a geometric category. In order to feed the first-layer geometry output to the second-layer classifiers, for scene/event categorization we form a vector with the predicted scores of all pixels; for the other tasks we use the 3-dimensional vector with each element representing the average scores for the corresponding pixel/patch/bounding-box.

6 Experiments and Results

6.1 Experimental Setting

The proposed FE-CCM model is a unified model which jointly optimizes for all the sub-tasks. We believe this is a powerful algorithm in that, while independent efforts towards each sub-task have led to state-of-the-art algorithms that require intricate modeling for that specific sub-task, the proposed approach is a unified model which can beat the state-of-the-art performance in each sub-task and, can be seamlessly applied across different applications.

We evaluate our proposed method on combining six tasks introduced in Section 5. In our experiment, the training of FE-CCM takes 4-5 iterations. For each of the sub-tasks in each of the domains, we evaluate our performance on the standard dataset for that sub-task (and compare against the specifically designed state-of-the-art algorithm for that dataset). Note that, with such disjoint yet practical datasets, no image would have ground truth available for more than one task. Our model handles this well.

In experiment we evaluate the following algorithms as shown in Table 1,

- Base model: Our implementation (Section 5) of each sub-task, which serves as a base model for our **FE-CCM**. (The base model uses less information than state-of-the-art algorithms for some sub-tasks.)
- All-features-direct: A classifier that takes all the features of all sub-tasks, appends them together, and builds a separate classifier for each sub-task.
- State-of-the-art model: The state-of-the-art algorithm for each sub-task respectively on that specific dataset.
- CCM: The cascaded classifier model by Heitz et al. [11], which we re-implement for six sub-tasks.
- FE-CCM (unified): This is our proposed model. Note that this is *one single model* which maximizes the joint likelihood of all the sub-tasks.
- FE-CCM (one goal): In this case, we have only one sub-task instantiated on the second layer, and the goal is to optimize the outputs of that sub-task. We train a specific one-goal FE-CCM for each sub-task.

TABLE 1

Summary of results for the SIX vision tasks. Our method improves performance in every single task. (Note: Bold face corresponds to our model performing equally with or better than state-of-the-art.)

Model	Event Categorization (% Accuracy)	Depth Estimation (RMSE in m)	Scene Categorization (% Accuracy)	Saliency Detection (% Accuracy)	Geometric Labeling (% Accuracy)	Object detection				
						Car	Person	Horse	Cow	Mean
						(% Average precision)				
Images in testset	1579	400	2688	1000	300	2686				
Chance	22.5	24.6	22.5	50	33.3	-	-	-	-	-
Our base-model	71.8 (±0.8)	16.7 (±0.4)	83.8 (±0.2)	85.2 (±0.2)	86.2 (±0.2)	62.4	36.3	39.0	39.9	44.4
All-features-direct	72.7 (±0.8)	16.4 (±0.4)	83.8 (±0.4)	85.7 (±0.2)	87.0 (±0.6)	62.3	36.8	38.8	40.0	44.5
State-of-the-art model (reported)	73.4 Li [46]	16.7 (MRF) [4] Saxena [59]	83.8 Torralba [58]	82.5 (±0.2) Achanta [62]	88.1 Hoiem [33]	61.5	36.3	39.2	40.7	44.4
						Felzenswalb et. al. [38] (base)				
CCM [11] (our implementation)	73.3 (±1.6)	16.4 (±0.4)	83.8 (±0.6)	85.6 (±0.2)	87.0 (±0.6)	62.2	37.0	38.8	40.1	44.5
FE-CCM (unified)	**74.3 (±0.6)**	**15.5 (±0.2)**	**85.9 (±0.3)**	**86.2 (±0.2)**	**88.6 (±0.2)**	**63.2**	**37.6**	**40.1**	40.5	**45.4**
FE-CCM (one goal)	**74.2 (±0.8)**	**15.3 (±0.4)**	**85.8 (±0.5)**	**87.1 (±0.2)**	**88.6 (±0.3)**	**63.2**	**37.9**	**40.1**	**40.7**	**45.5**
FE-CCM (target specific)	**74.7 (±0.6)**	**15.2 (±0.2)**	**86.1 (±0.2)**	**87.6 (±0.2)**	**88.9 (±0.2)**	**63.2**	**38.0**	**40.1**	**40.7**	**45.5**

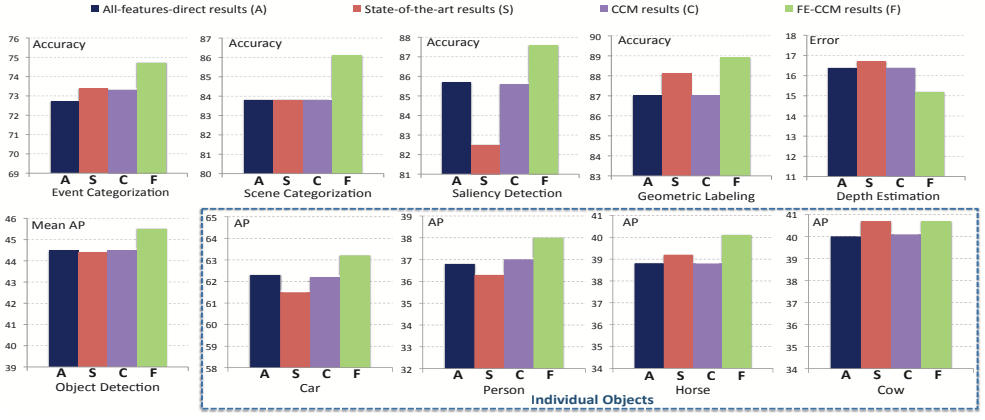

Fig. 3. Results for the six tasks in scene understanding. Top: the performance for event categorization, scene categorization, saliency detection, geometric labeling, and depth estimation. Bottom: the average performance for object detection and the performance for the detection of individual object categories: car, person, horse, and cow. Each figure compares four methods: all-features-direct method, state-of-the-art methods, CCM, and the proposed FE-CCM method.

- FE-CCM (target specific): In this case, we train a specific FE-CCM for each sub-task, by using cross-validation to estimate π_j's in Equation 23. Different values for π_j's result in different parameters learned for each FE-CCM.

Note that both CCM and All-features-direct use information from all sub-tasks, and state-of-the-art models also use carefully designed models that implicitly capture information from the other sub-tasks.

6.2 Datasets

The datasets used are mentioned in Section 5, and the number of test images in each dataset is shown in Table 1. For each dataset we use the same number of training images as the state-of-the-art algorithm (for comparison). We perform 6-fold cross validation on the whole model with 5 of 6 sub-tasks to evaluate the performance on each task. We do not do cross-validation on object detection as it is standard on the PASCAL 2006 [63] dataset (1277 train and 2686 test images respectively).

6.3 Results

To quantitatively evaluate our method for each of the sub-tasks, we consider the metrics appropriate to each of the six tasks in Section 5. Table 1 and Figure 3 show that FE-CCM not only beats state of the art in *all* the tasks but also does it jointly as *one single unified model*.

In detail, we see that all-features-direct improves over the base model because it uses features from all the tasks. The state-of-the-art classifiers improve on the base model by explicitly hand-designing the task specific probabilistic model [46, 59] or by using adhoc methods to implicitly use information from other tasks [33]. Our FE-CCM model, which is a single model that was not given any manually designed task-specific insight, achieves a more significant improvement over the base model.

We also compare the three instantiations of FE-CCM in Table 1 (the last three rows). We observe that the target-specific FE-CCM achieves the best performance, by selecting a set of π_j's to optimize for each task independently. Though the unified FE-CCM achieves slightly worse performance, it jointly optmizes for all the tasks by training only one set of parameters. The performance of one-goal FE-CCM is less stable compared to the other two instantiations. It is mainly because the first-layer classifiers only gain feedback from the specific task on the second layer in one-goal FE-CCM, which easily causes overfitting.

We note that our target-specific FE-CCM, which is optimized for each task independently and achieves the best performance, is a more fair comparison to the state-of-the-art because each state-of-the-art model is trained specifically for the respective task. Furthermore, Figure 3 shows the results for CCM (which is a cascade without feedback information) and all-features-direct (which uses features

4. The state-of-the-art method for depth estimation in [59] follows a slightly different testing procedure. In that case, our target-specific FE-CCM method achieves $RMSE = 15.3$.

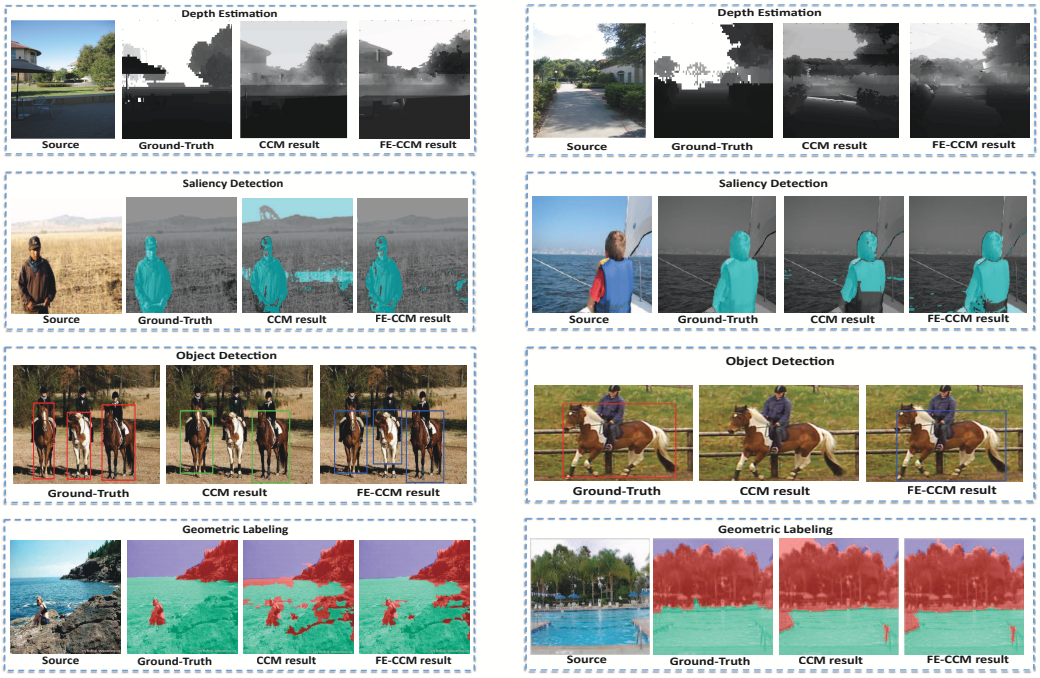

Fig. 4. Results showing improvement using the proposed model. From top to bottom: Depth estimation, Saliency detection, Object detection, Geometric labeling. All depth maps in depth estimation are at the same scale (black means near and white means far); Salient region in saliency detection are indicated in cyan; Geometric labeling: Green=Support, Blue=Sky and Red=Vertical (Best viewed in color).

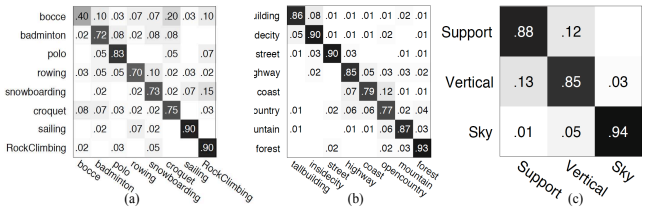

Fig. 5. Confusion matrix for (a) Event categorization; (b) Scene categorization; (c) Geometric labeling. All the results are gained with the proposed FE-CCM method. The average accuracy achieved by the proposed FE-CCM model outperforms the state-of-the-art methods for each of these tasks, as listed in Table 1.

from all the tasks). This indicates that the improvement is strictly due to the proposed feedback and not just because of having more information.

We show some visual improvements due to the proposed FE-CCM in Figure 4. In comparison to CCM, FE-CCM leads to better depth estimation of the sky and the ground, and it leads to better coverage and accurate labeling of the salient region in the image, and it also leads to better geometric labeling and object detection. Figure 5 also provides the confusion matrices for the three tasks: scene categorization, event categorization, geometric labeling.

Figure 6 provides scatter plots of the performance difference for each image between the unified FE-CCM method and the all-features-direct method, respectively for the tasks of geometric labeling, saliency detection, and depth estimation. We note that for all three tasks, the unified FE-CCM outperforms the all-features-direct method on most images. For geometric labeling and saliency detection, the improvement from the unified FE-CCM method is mainly due to large improvements on some images. For depth estimation, the improvement is scattered over many images.

The cause of improvement. We have shown improvements of FE-CCM in Table 1 under the situation of heterogeneous

TABLE 2

Summary of results for combining scene categorization and object detection, with partially-labeled datasets and fully-labeled datasets.

Model	Scene Categorization (% accuracy) partial-labeled / full-labeled	Object Detection (% mean AP) partial-labeled / full-labeled
Our base-model	45.6 / 47.5	67.6 / 70.7
All features direct	46.8 / 49.1	71.2 / 72.5
CCM [11]	50.8 / 52.3	74.0 / 76.1
FE-CCM (unified)	**54.2 / 54.8**	**77.5 / 77.9**

datasets. The improvement can be caused by one or both of the following reasons: (1) the feedback process finds better error modes for the first-layer classifiers; (2) the feedback generates additional "labels" to retrain the first-layer classifiers. In order to analyze this, we consider the two tasks of scene recognition and object detection on the DS1 dataset in [11], which contains ground-truth labels for both the tasks. We compare the various methods under two settings: (1) train with the fully-labeled data; (2) train with only the scene labels for one half of the training data and only the object labels for the second half. Table 2 compares the performance of training with partially-labeled datasets and the performance of different methods under these two settings. The experiments are performed using 5-fold cross validation. The unified FE-CCM method outperforms the other methods under both partially-labeled and fully-labeled situations. We note that all methods listed perform better when full labels are provided. In fact, FE-CCM achieves close performance in both settings. We also note that the FE-CCM method trained with partially-labeled datasets outperforms the CCM method trained with fully-labeled datasets, which indicates that the improvement achieved by the FE-CCM method is not simply from generating more labels for training the first-layer classifiers, but also due to finding useful modes for the first-layer classifiers.

Figure 7 illustrates the first-layer outputs of a test image, respectively at initialization and at the 5^{th} iteration. Our

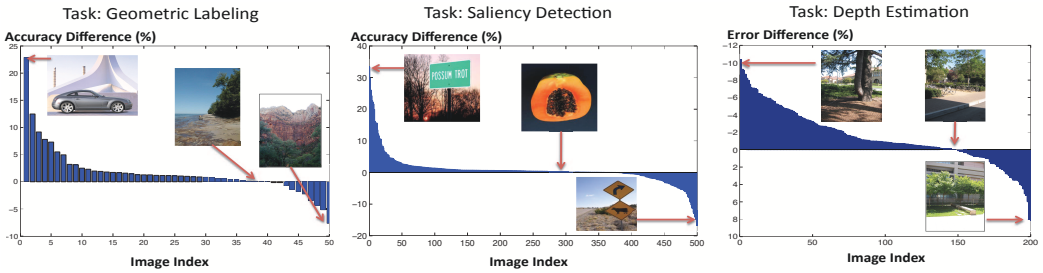

Fig. 6. Performance difference between the proposed unified FE-CCM method and the all-features-direct method for each test image, respectively for the tasks of geometric labeling, saliency detection, depth estimation, on one of the cross-validation folds.

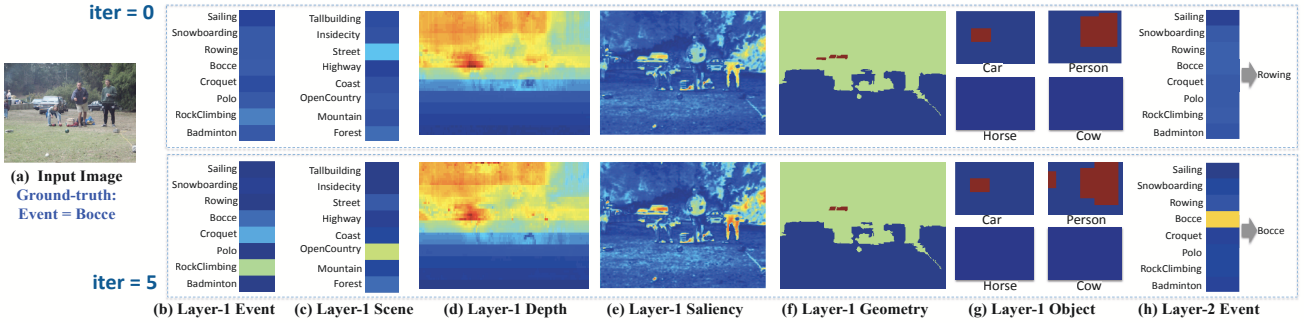

Fig. 7. Illustration of the FE-CCM first-layer outputs for a single image. (a) the input image from the sports-event dataset. Its groundtruth event label is "Bocce". (b-g)Outputs of the first-layer classifiers, at initialization (top row) and at the 5^{th} iteration (bottom row). (h) Outputs of the second-layer event classifier. Note that at initialization the first-layer classifiers are trained using ground-truth labels, i.e. the same as CCM. In (b)(c)(d)(e)(h), Red=High-value, Blue=low-value. In (f), Blue=Ground, Green=Vertical, Red=Sky. In (g) Red=Object Presence. (Best viewed in color.)

initialization is the same as CCM, i.e., using ground-truth labels to train the first-layer classifiers. We note that with feedback, the first-layer output shifts to focus on more meaningful modes, e.g., At initialization, the event classifier has widespread confusion with other categories. With feedback, the event classifier turns to be confused with only the 'rock-climbing' and 'croquet' events which are more similar to 'bocce'. Moreover, the first-layer scene, depth, and object classifiers also give more meaningful predictions while trained with feedback. With better first-layer predictions, our FE-CCM correctly classifies the event as 'bocce', while CCM misclassifies it as 'rowing'.

6.4 Discussion

FE-CCM allows each classifier in the second layer to learn which information from the other first-layer sub-tasks is useful, and this can be seen in the learned weights Ω for the second-layer. We provide a visualization of the weights for the six vision tasks in Figure 8(a). We see that the model agrees with our intuitions that large weights are assigned to the outputs of the same task from the first layer classifier (see the large weights assigned to the diagonals in the categorization tasks), though saliency detection is an exception which depends more on its original features (not shown here) and the geometric labeling output. We also observe that the weights are sparse. This is an advantage of our approach since the algorithm automatically figures out which outputs from the first level classifiers are useful for the second level classifier to achieve the best performance.

Figure 8(b) provides a closer look to the positive weights given to the various outputs for a second-level geometric classifier. We observe that large positive weights are assigned to "mountain", "forest", "tall building", etc. for supporting the geometric class "vertical", and similarly "coast", "sailing" and "depth" for supporting the "sky" class. These illustrate some of the relationships the model learns automatically without any manual intricate modeling.

Figure 8(c) visualizes the weights given to the depth attributes (first-layer depth outputs) for the task of event categorization. Figure 8(d) shows the same for the task of scene categorization. We see that the depth plays an important role in these tasks. In Figure 8(c), we observe that most event categories rely on the middle part of the image, where the main objects of the event are often located. 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 the others. Not surprisingly, our method had automatically figured this out (see Figure 8(d)).

Stability of the FE-CCM algorithm: In this paper, we have presented results for six sub-tasks. In order to find out how our method scales with different combination and number of sub-tasks, we have tried several combinations, and in each case we get consistent improvement in each sub-task. For example, in our preliminary experiments, we combined depth estimation and scene categorization and our reduction in error are 12.0% and 13.2% respectively. Combining scene categorization and object detection gives us 15.4% and 10.2% respective improvements (Table 2). We then combined four tasks: event categorization, scene categorization, depth estimation, and saliency detection, and got improvements in all these sub-tasks [22]. Finally, we also combined different tasks for robotic applications, and the performance improvement was similar.

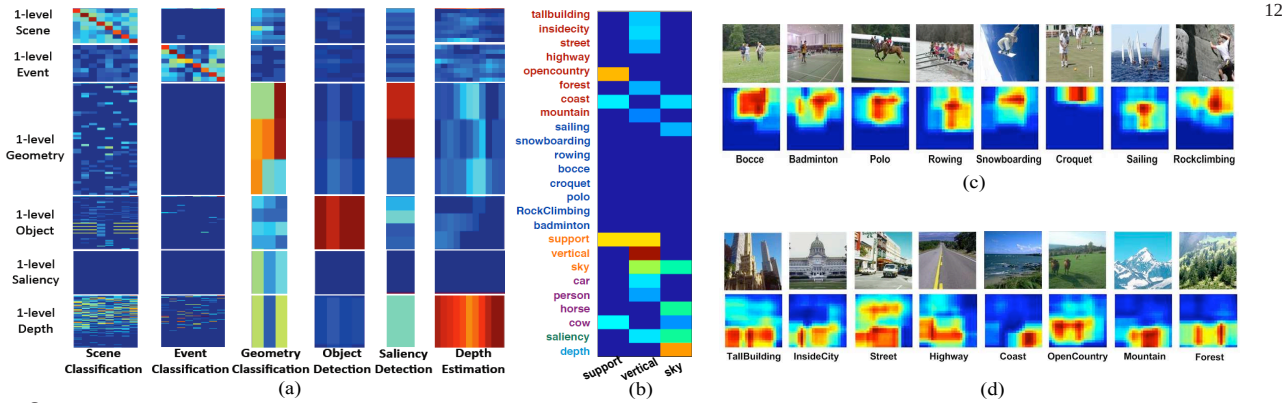

Fig. 8. (a) The *absolute* values of the weight vectors for second-level classifiers, i.e. ω. Each column shows the contribution of the various tasks towards a certain task. (b) Detailed illustration of the *positive* values in the weight vector for a second-level geometric classifier. (c)(d) Illustration of the importance of depths in different regions for predicting different events (c) and scenes (d). An example image for each class is also shown above the map of the weights. (Note: Blue is low and Red is high. Best viewed in Color).

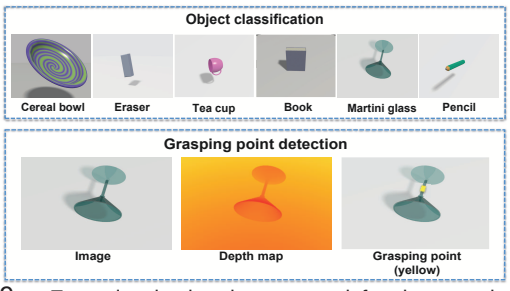

Fig. 9. Examples in the dataset used for the grasping robot experiments. The two tasks considered were a six-class, object classification task and grasping point detection task.

7 Robotic Applications

In order to show the applicability of our FE-CCM to different scene understanding domains, we also used the proposed method in multiple robotic applications.

7.1 Robotic Grasping

Given an image and a depthmap (Figure 9), the goal of the learning algorithm in a grasping robot is to select a point to grasp the object (this location is called the grasp point, [66]). It turns out that different categories of objects demand different strategies for grasping. In prior work, Saxena et al. [66, 67] did not use object category information for grasping. In this work, we use our FE-CCM to combine object classification and grasping point detection.

Implementation: We work with the labeled synthetic dataset by Saxena et al. [66] which spans 6 object categories and also includes an aligned pixel level depth map for each image, as shown in Figure 9. The six object categories include spherically symmetric objects such as cerealbowl, rectangular objects such as eraser, martini glass, books, cups and long objects such as pencil.

For grasp point detection, we compute image and depthmap features at each point in the image (using codes given by [66]). The features describe the response of the image and the depth map to a bank of filters (similar to Make3D) while also capturing information from the neighboring grid elements. We then use a regression over the features. The output is a confidence score for each point being a good grasping point. In an image, we pick the point with the highest score as the grasping point.

For object detection, we use a logistic classifier to perform the classification. The output of the classifier is a 6-dimensional vector representing the log-odds score for each category. The final classification is performed by assigning the image to the category with the highest score.

TABLE 3
Summary of results for the the robotic grasping experiment. Our method improves performance in every single task.

Model	Graping point Detection (% accuracy)	Object Classification (% accuracy)
Images in testset	6000	1200
Chance	50	16.7
All features direct	87.7	45.8
Our base-model	87.7	45.8
CCM (Heitz et. al.)	90.5	49.5
FE-CCM	**92.2**	**49.7**

Fig. 10. Left: the grasping point detected by our algorithm. Right: Our robot grasping an object using our algorithm.

Results: We evaluate our algorithm on a dataset published in [66], and perform cross-validation to evaluate the performance on each task. We use 6000 images for grasping point detection (3000 for training and 3000 for testing) and 1200 images for object classification (600 for training and 600 for testing). Table 3 shows the results for our algorithm's ability to predict the grasping point, given an image and the depths observed by the robot using its sensors. We see that our FE-CCM obtains significantly better performance over all-features-direct and CCM (our implementation). Figure 10 shows an example of our robot grasping an object.

7.2 Object-finding Robot

Given an image, the goal of an object-finding robot is to find a desired object in a cluttered room. As we have discussed earlier, some types of scenes such as living room are more likely to have objects (e.g., shoes) than other types of scenes such as kitchen. Similarly, office scenes are more likely to contain tv-monitors than kitchen scenes. Furthermore, it is also intuitive that shoes are more likely to appear on the supportive surface such as floor, instead of the vertical surface such as the wall. Therefore, in this work, we use our FE-CCM to combine object detection with indoor scene categorization and geometric labeling.

Implementation: For scene categorization, we use the indoor scene subsets in the Cal-Scene Dataset [68] and

Fig. 11. Left: the shoe-finding robot, which has a camera to take photos of a scene. Right: the shoed detected using our algorithm.

classify an image into one of the four categories: bedroom, living room, kitchen and office. For geometric labeling, we use the Indoor Layout Data [69] and assign each pixel to one of three geometry classes: ground, wall and ceiling. We use the same features and classifiers for scene categorization as in Section 5.

For object detection, we use the PASCAL 2007 Dataset [70] and our own shoe dataset to learn detectors for four object categories: shoe, dining table, tv-monitor, and sofa. We first use the part-based object detection algorithm in [38] to create candidate windows, and then use the same classifiers as described in Section 5.

Results: We use this method to build a shoe-finding robot, as shown on Figure 11-left. With a limited number of training images (86 positive images in our case), it is hard to train a robust shoe detector to find a shoe far away from the camera. However, using our FE-CCM model, the robot learns to leverage the other tasks and performs more robust shoe detection. Figure 11-right shows a successful detection. For more details and videos, please see [71].

8 Conclusions

We propose a method for combining existing classifiers for different but related tasks in scene understanding. We only consider the individual classifiers as a 'black-box' (thus not needing to know the inner workings of the classifier) and propose learning techniques for combining them (thus not needing to know how to combine the tasks). Our method introduces feedback in the training process from the later stage to the earlier one, so that a later classifier can provide the earlier classifiers information about what error modes to focus on, or what can be ignored without hurting the joint performance.

Our extensive experiments show that our unified model (a single FE-CCM trained for all the sub-tasks) improves performance significantly across *all* the sub-tasks considered over the respective state-of-the-art classifiers. We show that this was the result of our feedback process. The classifier actually learns meaningful relationships between the tasks automatically. We believe that this is a small step towards holistic scene understanding.

Acknowledgments

We thank Anish Nahar, Matthew Cong, TP Wong, Norris Xu, and Colin Ponce for help with the robotic experiments. We also thank Daphne Koller for useful discussions.

References

[1] S. Kumar and M. Hebert, "A hierarchical field framework for unified context-based classification," in *ICCV*, 2005.
[2] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *PAMI*, vol. 30, no. 5, 2009.
[3] D. Hoiem, A. A. Efros, and M. Hebert, "Closing the loop on scene interpretation," in *CVPR*, 2008.
[4] L.-J. Li, R. Socher, and L. Fei-Fei, "Towards total scene understanding: Classification, annotation and segmentation in an automatic framework," in *CVPR*, 2009.
[5] E. B. Sudderth, A. Torralba, W. T. Freeman, and A. S. Willsky, "Depth from familiar objects: A hierarchical model for 3d scenes," in *CVPR*, 2006.
[6] C. Sutton and A. McCallum, "Joint parsing and semantic role labeling," in *CoNLL*, 2005.
[7] D. Parikh, C. Zitnick, and T. Chen, "From appearance to context-based recognition: Dense labeling in small images," *CVPR*, 2008.
[8] A. Toshev, B. Taskar, and K. Daniilidis, "Object detection via boundary structure segmentation," in *CVPR*, 2010.
[9] A. Agarwal and B. Triggs, "Monocular human motion capture with a mixture of regressors," in *CVPR Workshop on Vision for HCI*, 2005.
[10] A. Saxena, J. Schulte, and A. Y. Ng, "Depth estimation using monocular and stereo cues," in *IJCAI*, 2007.
[11] G. Heitz, S. Gould, A. Saxena, and D. Koller, "Cascaded classification models: Combining models for holistic scene understanding." in *NIPS*, 2008.
[12] L. Hansen and P. Salamon, "Neural network ensembles," *PAMI*, vol. 12, no. 10, pp. 993–1001, 1990.
[13] Y. Freund and R. E. Schapire, "Cascaded neural networks based image classifier," in *ICASSP*, 1993.
[14] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *ICML*, 2008.
[15] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *EuroCOLT*, 1995.
[16] S. C. Brubaker, J. Wu, J. Sun, M. D. Mullin, and J. M. Rehg, "On the design of cascades of boosted ensembles for face detection," *IJCV*, vol. 77, no. 1-3, pp. 65–86, 2008.
[17] P. Viola and M. J. Jones, "Robust real-time face detection," *IJCV*, vol. 57, no. 2, pp. 137–154, 2004.
[18] M. Fink and P. Perona, "Mutual boosting for contextual inference," in *In NIPS*, 2004.
[19] A. Torralba, K. Murphy, and W. Freeman, "Contextual models for object detection using boosted random fields," in *In NIPS*, 2005.
[20] Z. Tu, "Auto-context and its application to high-level vision tasks," in *CVPR*, 2008.
[21] C. Li, A. Kowdle, A. Saxena, and T. Chen, "Feedback enabled cascaded classication models for scene understanding," in *NIPS*, 2010.
[22] A. Kowdle, C. Li, A. Saxena, and T. Chen, "A generic model to compose vision modules for holistic scene understanding," in *ECCV Workshop on Parts and Attributes*, 2010.
[23] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, "On combining classifiers," *PAMI*, vol. 20, pp. 226–239, 1998.
[24] C. Li, A. Saxena, and T. Chen, "θ-mrf:capturing spatial and semantic structure in the parameters for scene understanding," in *NIPS*, 2011.
[25] R. K. Ando and T. Zhang, "A framework for learning predictive structures from multiple tasks and unlabeled data," *J. Mach. Learn. Res.*, vol. 6, pp. 1817–1853, December 2005.
[26] I. Tsochantaridis, T. Hofmann, and T. Joachims, "Support vector machine learning for interdependent and structured output spaces," in *ICML*, 2004.
[27] B. Taskar, C. Guestrin, and D. Koller, "Max-margin markov networks," in *NIPS*, 2003.
[28] H. Koppula, A. Anand, T. Joachims, and A. Saxena, "Semantic labeling of 3d point clouds for indoor scenes," in *Neural Information Processing Systems (NIPS)*, 2011.
[29] A. Quattoni, M. Collins, and T. Darrell, "Conditional random fields for object recognition," in *In NIPS*, 2004.
[30] C.-N. J. Yu and T. Joachims, "Learning structural svms with latent variables," in *ICML*, 2009.
[31] A. Torralba, "Contextual priming for object detection," *Int. J. Comput. Vision*, vol. 53, no. 2, pp. 169–191, 2003.
[32] A. Torralba and A. Oliva, "Depth estimation from image structure," *IEEE PAMI*, vol. 24, pp. 1226–1238, September 2002.
[33] D. Hoiem, A. A. Efros, and M. Hebert, "Putting objects in perspective," *IJCV*, 2008.
[34] D. Park, D. Ramanan, and C. Fowlkes, "Multiresolution models for object detection," in *ECCV*, 2010.
[35] G. Heitz and D. Koller, "Learning spatial context: Using stuff to find things," in *ECCV*, 2008.
[36] J. Lim, P. Arbel andez, C. Gu, and J. Malik, "Context by region ancestry," in *ICCV*, 2009.
[37] S. Kumar and M. Hebert, "A hierarchical field framework for unified

context-based classification," in *ICCV*, 2005.
[38] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *PAMI*, 2009.
[39] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie, "Objects in context," in *ICCV*, 2007.
[40] D. Parikh, C. Zitnick, and T. Chen, "From appearance to context-based recognition: Dense labeling in small images," in *CVPR*, 2008.
[41] C. Desai, D. Ramanan, and C. Fowlkes, "Discriminative models for multi-class object layout," in *ICCV*, 2009.
[42] B. Yao and L. Fei-Fei, "Modeling mutual context of object and human pose in human-object interaction activities," in *CVPR*, 2010.
[43] C. Galleguillos, B. McFee, S. Belongie, and G. Lanckriet, "Multi-class object localization by combining local contextual interactions," in *CVPR*, 2010.
[44] S. Divvala, D. Hoiem, J. Hays, A. Efros, and M. Hebert, "An empirical study of context in object detection," in *CVPR*, 2009.
[45] M. Blaschko and C. Lampert, "Object localization with global and local context kernels," in *BMVC*, 2009.
[46] L. Li and L. Fei-Fei, "What, where and who? classifying event by scene and object recognition," in *ICCV*, 2007.
[47] Y. Bengio and Y. LeCun, "Scaling learning algorithms towards ai," in *Large-Scale Kernel Machines*, 2007.
[48] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, pp. 41–75, 1997.
[49] Y. LeCun, L. Bottou, G. Orr, and K. Muller, "Efficient backprop," in *Neural Networks: Tricks of the trade*, G. Orr and M. K., Eds. Springer, 1998.
[50] I. Goodfellow, Q. Le, A. Saxena, H. Lee, and A. Ng, "Measuring invariances in deep networks," in *NIPS*, 2009.
[51] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," in *N. Comp*, 2006.
[52] M. Zeiler, D. Krishnan, G. Taylor, and R. Fergus, "Deconvolutional networks," in *CVPR*, 2010.
[53] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *J of Royal Stat. Soc., Series B*, vol. 39, no. 1, pp. 1–38, 1977.
[54] R. Neal and G. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," *Learning in graphical models*, vol. 89, pp. 355–368, 1998.
[55] M. Gibbs and D. Mackay, "Variational gaussian process classifiers," *Neural Networks, IEEE Trans*, 2000.
[56] J. Mairal, M. Leordeanu, F. Bach, M. Hebert, and J. Ponce, "Discriminative sparse image models for class-specific edge detection and image interpretation," in *ECCV*, 2008.
[57] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *IJCV*, vol. 42, pp. 145–175, 2001.
[58] A. Torralba and A. Oliva, "MIT outdoor scene dataset," http://people.csail.mit.edu/torralba/code/spatialenvelope/index.html.
[59] A. Saxena, S. H. Chung, and A. Y. Ng, "3-d depth reconstruction from a single still image," *IJCV*, vol. 76, 2007.
[60] A. Saxena, S. Chung, and A. Ng, "Learning depth from single monocular images," in *NIPS*, 2005.
[61] A. Torralba, A. Oliva, M. S. Castelhano, and J. M. Henderson, "Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search." *Psychol Rev*, vol. 113, no. 4, 2006.
[62] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk, "Frequency-tuned Salient Region Detection," in *CVPR*, 2009.
[63] M. Everingham, A. Zisserman, C. K. I. Williams, and L. Van Gool, "The PASCAL VOC2006 Results." [Online]. Available: http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf
[64] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Discriminatively trained deformable part models, release 3," http://people.cs.uchicago.edu/∼pff/latent-release3/.
[65] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *CVPR*, 2005.
[66] A. Saxena, J. Driemeyer, J. Kearns, and A. Y. Ng, "Robotic grasping of novel objects," in *NIPS*, 2006.
[67] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *IJRR*, vol. 27, no. 2, pp. 157–173, 2008.
[68] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," *CVPR*, 2005.
[69] V. Hedau, D. Hoiem, and D. Forsyth, "Recovering the spatial layout of cluttered rooms," in *ICCV*, 2009.
[70] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.
[71] C. Li, T. Wong, N. Xu, and A. Saxena, "FECCM for scene unverstanding: Helping the robot to learn multiple tasks," in *Video track ICRA. Online URL: http://chenlab.ece.cornell.edu/projects/FECCM/*, 2011.

Congcong Li is a Ph.D. candidate in the School of Electrical and Computer Engineering, Cornell University. She received her B.E. in 2005 and M.S. in 2007 from Tsinghua University, China. She started her Ph.D. study in Carnegie Mellon University, from 2007 and moved to Cornell University, in 2009. Her research interests include computer vision, machine learning, and image analysis.

Adarsh Kowdle is a Ph.D. student in the School of Electrical and Computer Engineering, Cornell University. He received his B.E. from Rashtreeya Vidyalaya College of Engineering (India) in 2007. He worked with Ittiam Systems (India) from June 2007 to July 2008. Before starting his Ph.D. in Cornell University in 2009, he spent a semester at Carnegie Mellon University. His research interests include interactive computer vision algorithms and machine learning.

Ashutosh Saxena is an assistant professor in computer science department at Cornell University. His research interests include machine learning, robotics and computer vision. He received his MS in 2006 and Ph.D. in 2009 from Stanford University, and his B.Tech. in 2004 from Indian Institute of Technology (IIT) Kanpur.

Prof. Saxena has developed Make3D (http://make3d.cs.cornell.edu), an algorithm that converts a single photograph into a 3D model. Tens of thousands of users used this technology to convert their pictures to 3D. He has also developed algorithms that enable robots to perform household chores such as load and unload items from a dishwasher. His work has received substantial amount of attention in popular press, including the front-page of New York Times, BBC, ABC, New Scientist Discovery Science, and Wired Magazine. He has won best paper awards in 3DRR and IEEE ACE. He was also a recipient of National Talent Scholar award in India, and Sloan research fellowship in 2011.

Tsuhan Chen received the B.S. degree in electrical engineering from the National Taiwan University, Taipei, in 1987 and the M.S. and Ph.D. degrees in electrical engineering from the California Institute of Technology, Pasadena, in 1990 and 1993, respectively. He has been with the School of Electrical and Computer Engineering, Cornell University, Ithaca, NewYork, since January 2009, where he is Professor and Director. From October 1997 to December 2008, he was with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, as Professor and Associate Department Head. From August 1993 to October 1997,he worked at AT&T Bell Laboratories, Holmdel, NJ. He coedited a book titled Multimedia Systems, Standards, and Networks (Marcel Dekker, 2000).

Prof. Chen served as the Editor-in-Chief for the IEEE Transactions on Multimedia in 2002 – 2004. He also served in the Editorial Board of IEEE Signal Processing Magazine and as Associate Editor for IEEE Transactions on Circuits and Systems for Video Technology, IEEE Transactions on Image Processing, IEEE Transactions on Signal Processing, and IEEE Transactions on Multimedia. He received the Charles Wilts Prize at the California Institute of Technology in 1993. He was a recipient of the National Science Foundation Career Award, from 2000 to 2003. He received the Benjamin Richard Teare Teaching Award in 2006, and the Eta Kappa Nu Award for Outstanding Faculty Teaching in 2007. He was elected to the Board of Governors, IEEE Signal Processing Society, 2007 – 2009, and a Distinguished Lecturer, IEEE Signal Processing Society, 2007 – 2008. He is a member of the Phi Tau Phi Scholastic Honor Society.