Towards Holistic Scene Understanding: Feedback Enabled Cascaded Classification Models

Congcong Li, *Student Member, IEEE,* Adarsh Kowdle, *Student Member, IEEE,* Ashutosh Saxena, *Member, IEEE,* and 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 maximizes the joint likelihood of 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 achieves advanced performance in two robotic applications including the object-grasping robot and the object-finding robot.

Index Terms—Scene understanding, Classification.

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 which 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 application domains.

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 most 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 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 depth well

- *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*

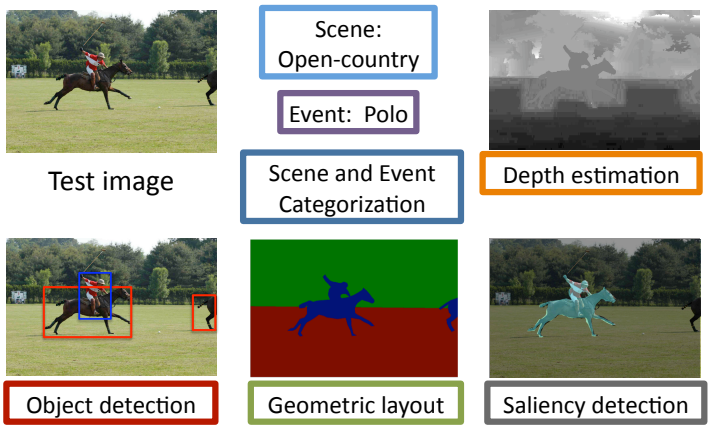


Fig. 1. Given a test image, 'Holistic Scene Understanding' corresponds to inferring the labels for all possible scene understanding dimensions. In our work, we infer labels corresponding to, scene categorization, event categorization, depth estimation (Black = close, white = far), object detection (red = horse, blue = person), geometric layout (green = vertical-porous, red = horizontal, blue = vertical) and saliency detection (cyan = salient) as shown above and achieve this jointly using one unified model. The idea of a unified model for scene understanding is illustrated here. It is clear to see that different tasks can 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. (Best viewed in color).

in that region by sacrificing some regions in the bottom may help to correctly classify an image. In detail, we do so by jointly maximizing the likelihood of all the tasks; the outputs of the first layers are treated as latent variables and training is done by 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 the 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 some 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 give an overview of our approach in Section 3. We describe our FE-CCM method in Section 4 followed by the discussion about handling heterogeneous datasets in Section 5. We provide the implementation details of the classifiers in Section 6. We present the experiments and results in Section 7 and some robotic applications in Section 8. We finally conclude in Section 9.

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 are interested in answering several different questions: What objects there are 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 $y_i \in S_i$ for the i^{th} sub-task. If we are considering depth prediction (see Figure 1), then the label would be $y_1 \in S_1 = \mathbb{R}_+^{100\times100}$ for continuous values of depth in a 100×100 output. For scene categorization, we will have $y_2 \in S_2 = \{1, \ldots, K\}$ for K scene classes. And so on.

If we have n sub-tasks, then we would have to produce an output as:

$$y = \{y_1, \ldots, y_n\} \in S_1 \times S_2 \ldots \times S_n.$$

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

2.2 Related Work

Cascaded classifiers for a single task. The idea of using information from related tasks to improve the performance of the task in question has been studied in various fields of machine learning and vision. The idea of cascading layers of classifiers to aid the final 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 hidden layer [12–14]. However, it is often hard to train neural networks and gain an insight into its operation, thus 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]. However,

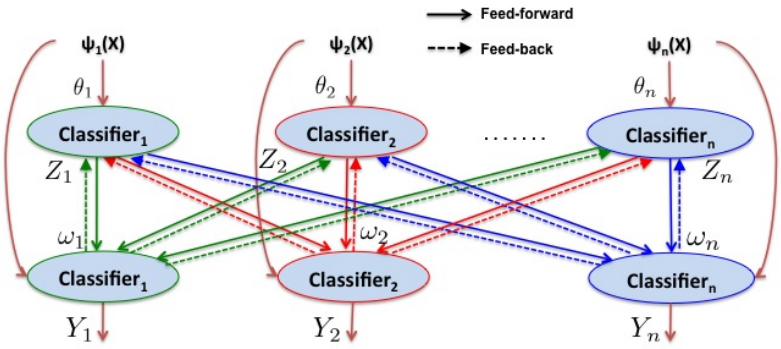


Fig. 2. The proposed Feed-back enabled cascaded classification model (FE-CCM) for combining related classifiers. ($\forall i \in \{1, 2, \ldots, n\}$ $\Psi_i(X)$ = Features corresponding to $Classifier_i$ extracted from image X, Z_i = Output of the $Classifier_i$ in the first stage parameterized by θ_i, 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. (Note that different colors of lines are used only to make the figure more readable)

unlike the CCM framework which focuses on contextual benefits, the motivation in these cases was computational efficiency. Tu [18] used pixel-level label maps to learn a contextual model for pixel-level labeling through a cascaded classifier approach, but such works consider only the interactions between labels of the same type.

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 [19]. 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.

Graphical Models for combining tasks. While the methods discussed above combine classifiers to predict the *same* labels, there is a group of works that combine classifiers, and use them as components in large systems. Kumar and Hebert [4] developed a large MRF-based probabilistic model to link multi-class segmentation and object detection. Similar efforts have been made in the field of natural language processing. Sutton and McCallum [5] combined a parsing model with a semantic role labeling model into a unified probabilistic framework that solved both simultaneously. However, it is hard to fit existing state-of-the-art classifiers into these technically-sound probabilistic representations because they require knowledge of the inner workings of the individual classifiers.

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., [6–8]). Saxena et. al. manually designed the terms in an MRF to combine depth estimation with object detection [9] and stereo cues [10]. Sudderth et al. [3] 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. [20, 21] use the statistics of low-level features across the entire scene to prime object detection or help depth estimation. Hoiem et al. [22] use 3D scene information to provide priors on potential object locations. Park et al. [23] use the ground plane estimation as contextual information for pedestrian detection. Many works also model context to capture the local interactions between neighboring regions [24–26], objects [27–31], or both [32–34]. 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. [1] 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. [2, 35] 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.

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 [36] for a nice overview of deep learning architectures and Caruana [37] for multitask learning with shared representation. While efficient back-propagation methods like [38] 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., [39–41]) 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. [40] 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.

Structured Learning. Structured learning algorithms (e.g., [42, 43]) could also be a viable option for the setting of combining multiple tasks. However, these methods require a fully-labeled dataset which is often not available in vision tasks. Furthermore, they also require considerable effort in designing the loss function, which becomes extremely difficult when we consider multiple sub-tasks and the combined outputs become highly complex.

3 Our Approach

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

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

$$\textit{Inference} : \hat{Y}_i = \underset{Y_i}{\text{optimize}}\, f_{infer}(X, Y_i; \theta_i^*) \quad (2)$$

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 different 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.

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.

Notation: We consider related subtasks denoted by Classifier_i, where $i \in \{1, 2, \ldots, n\}$ for a total of n tasks (Figure 2). We describe the notations used in this paper as follows:

1. 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.

$\Psi_i(X)$	Features corresponding to Classifier_i extracted from image X.
Z_i, $\mathcal{Z}$	Z_i indicates output from the first layer Classifier_i. $\mathcal{Z}$ 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 k labeled pairs $\{X^{(k)}, Y_j^{(k)}\}$ in the training set. Γ represents all the labeled data.

4 Feedback Enabled Cascaded Classification Models

We will describe the learning and inference mechanisms for the proposed model in this section.

Given the classical setting of supervised learning, we model the conditional log likelihood of all the outputs for different 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 optimize

$$\log \prod_{X \in \Gamma} P(\mathcal{Y}|X; \Theta, \Omega) \quad (3)$$

where Θ, Ω are internal parameters of the black-box classifiers used in the proposed model.

In the following, we will explain how we introduce the first-layer outputs $\mathcal{Z}$ into the objective function above, and how the learning and inference procedures in our proposed model result in a sequence of steps, each involving original infernce and learning of the black-box classifiers in isolation.

4.1 Model Learning

Now we present our training procedure. During training, $Y_1, Y_2, \ldots, Y_n$ are all observed (because the ground-truth labels are available). However, $\mathcal{Z}$ (outputs of layer 1 and inputs to layer 2) are hidden, and this makes the training of each classifier as a black-box hard. 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 our classifiers to learn jointly, i.e., 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 expand Equation 3

as follows, using the independencies represented by the directed graphical model in Figure 2.

$$= \sum_{X \in \Gamma} \log \sum_{\mathcal{Z}} P(Y_1, \ldots, Y_n, \mathcal{Z} | X; \Theta, \Omega) \quad (4)$$

$$= \sum_{X \in \Gamma} \log \sum_{\mathcal{Z}} \prod_{i=1}^{n} P(Y_i | \Psi_i(X), \mathcal{Z}; \omega_i) P(Z_i | \Psi_i(X); \theta_i) \quad (5)$$

However, the summation inside the log makes it difficult to learn the parameters. Motivated by the Expectation Maximization [44] algorithm, we use an iterative algorithm where we first fix the latent variables Z_i's and learn the parameters in the first step (Feed-forward step), and estimate the latent variables Z_i's in the second step (Feed-back step). We then iterate between these two steps. The learning algorithm is summarized in Algorithm 1. While this algorithm is not guranteed to converge to the global maxima, in practice, we find it gives good results. The results of our algorithm are always better than [11] which in our formulation is equivalent to *fixing* the latent variables to ground-truth permanently (thus highlighting the impact of the feedback).

Algorithm 1 Learning

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

Do until convergence or maximum iteration:

{

Feed-foward step: Fix latent variables $\mathcal{Z}$, estimate the parameters Θ and Ω using Equation 9 and Equation 10.

Feed-back step: Fix the parameters Θ and Ω, compute latent variables $\mathcal{Z}$ using Equation 11.

}

Initialization: We initialize this process by setting the latent variables Z_i's to the groundtruth. 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 classfier and the classifiers in the second layer use the outputs of the first layer in addition to the original features.

Feed-forward Step: In this step, we estimate the parameters. We assume 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

$$\underset{\theta_1,\ldots,\theta_n,\omega_1,\ldots,\omega_n}{\text{maximize}} \sum_{X \in \Gamma} \log \prod_{i=1}^{n} P(Y_i | \Psi_i(X), \mathcal{Z}; \omega_i) P(Z_i | \Psi_i(X); \theta_i) \quad (6)$$

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:

$$\underset{\theta_i}{\text{maximize}} \sum_{X \in \Gamma} \log P(Z_i | \Psi_i(X); \theta_i) \quad (7)$$

$$\underset{\omega_i}{\text{maximize}} \sum_{X \in \Gamma} \log P(Y_i | \Psi_i(X), \mathcal{Z}; \omega_i) \quad (8)$$

Note that the optimization problem nicely breaks down into the sub-problems of training the individual classifier for the respective sub-tasks.

The maximization problem in Equation 7 is precisely the learning problem of the "blackbox classifier". The maximization problem in Equation 8 is also an instantiation of the original learning problem, but with more inputs. Therefore, we can use the learning method provided by the individual blackbox classifier (Equation 1).

$$\theta_i^* = \underset{\theta_i}{\text{optimize}}\, f_{learn}(\Psi_i(X), Y_i; \theta_i) \quad (9)$$

$$\omega_i^* = \underset{\omega_i}{\text{optimize}}\, f_{learn}([\Psi_i(X)\ \mathcal{Z}], Y_i; \omega_i) \quad (10)$$

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 that comes with the black-box).

Feed-back Step: In this second step, we will estimate the values of the latent variables Z_i's assuming that the parameters are fixed (and Y_i's are given because the ground-truth is available). 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.

We will 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, [45]). Using Equation 5, we get the following optimization problem for the feed-back step:

$$\underset{\mathcal{Z}}{\text{maximize}} \log P(Y_1, \ldots, Y_n, \mathcal{Z} | X; \theta_1, \ldots, \theta_n, \omega_1, \ldots, \omega_n) \Leftrightarrow$$

$$\underset{\mathcal{Z}}{\text{maximize}} \sum_{i=1}^{n} \Big(\log P(Z_i | \Psi_i(X); \theta_i) + \log P(Y_i | \Psi_i(X), \mathcal{Z}; \omega_i) \Big) \quad (11)$$

This maximization problem requires that we have access to the characterization of the individual black-box classifiers in a probabilistic form. While at first blush this may seem a difficult requirement, our method can even handle classifiers for which the log likelihood is not available. We can do this by taking the output of the previous classifiers and modeling their log-odds as a Gaussian (partly motivated by variational approximation methods [46]). Parameters of the Gaussians are empirically estimated when the actual model is not available.

In some cases, the classifier log-likelihoods in Equation 11 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 similar method). See Section 4.3 for more discussion on this.

4.2 Inference

Our inference procedure consists of two steps. We first maximize over hidden variable Z and then we maximize over Y. In detail, our optimal value of the output is given

by the following two steps: [2]

$$\hat{\mathcal{Z}} = \underset{\mathcal{Z}}{\operatorname{argmax}} \log P(\mathcal{Z}|X, \Theta) \tag{12}$$

$$\hat{\mathcal{Y}} = \underset{\mathcal{Y}}{\operatorname{argmax}} \log P(\mathcal{Y}|\hat{\mathcal{Z}}, X, \Omega) \tag{13}$$

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 12 and 13 are equal to the following:

$$\hat{Z}_i = \underset{Z_i}{\operatorname{argmax}} \log P(Z_i|X, \theta_i), i = 1, \ldots, n \tag{14}$$

$$\hat{Y}_i = \underset{Y_i}{\operatorname{argmax}} \log P(Y_i|\hat{\mathcal{Z}}, X, \omega_i), i = 1, \ldots, n \tag{15}$$

The inference algorithm is given in Algorithm 2. This approximate inference allows us to use the internal inference function (Equation 2) of the black-box classifiers without knowing its inner workings. It is tractable since and its complexity is no more than constant times the complexity of inference in the original classiers.

Algorithm 2 Inference

1. **Inference for first layer:**

   for $i = 1 : n$

   $$\hat{Z}_i = \underset{Z_i}{\text{optimize}}\, f_{infer}(\Psi_i(X), Z_i; \theta_i^*)$$

   end

2. **Inference for second layer:**

   for $i = 1 : n$

   $$\hat{Y}_i = \underset{Y_i}{\text{optimize}}\, f_{infer}([\Psi_i(X)\ \mathcal{Z}], Y_i; \omega_i^*)$$

   end

4.3 Modeling options

Solving Equation 11 in the feedback step depends on the modeling of $P(Z_i|\Psi_i(X); \theta_i)$ and $P(Y_i|\Psi_i(X), \mathcal{Z}; \omega_i)$. There are several methods to approach this problem depending on the situation:

1) Case 1: No insight into the vision problem, and no probabilistic interpretation of the original classifier for task i is available.
   In this case, we will append all the outputs $Z_1, \ldots, Z_n$ to the original feature vectors $\Psi_i(X^{(k)})$, and make a variational approximation on the output of the classifier for task i (i.e., approximating it as a Gaussian, [46]) to get:

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

   where $\hat{Y}_i$ is the actual output of the original classifier for the task i, α_i is the parameters for the approximation model.

   Sparsity: Note that the parameter set 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 using Laplace Prior $P(\alpha_i) \propto \exp(-\lambda||\alpha_i||_1)$. This prior is known to enforce sparsity in the parameters [47]. So Equation 16 is extended as follows.

   $$\underset{\alpha_i}{\text{minimize}} \sum_{k \in \Gamma} \left(\left|\left|\hat{Y}_i - \alpha_i^T[\Psi_i(X^{(k)}), \mathcal{Z}]\right|\right|_2^2 + \lambda\,|\alpha_i|\right) \tag{17}$$

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 $\mathcal{Z}$ to the original feature vectors $\Psi_i(X^{(k)})$, and solve problem Eqn 8 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 $P(Y_i|\Psi_i(X), \mathcal{Z})$.

In this paper, we show that even with Case 1, we get uniformly better results across all the sub-tasks. We believe that, if for a particular problem, one has insights into the vision problem, one could use Case 3 and achieve even better results than what we present here.

4.4 Model properties

In this section, we summarize the properties of our model:

- Our method has the ability to compose the individual classifiers without information on their inner workings. I.e., use state-of-the-art classifiers as a "black-box" (Section 4.1 and Section 4.2).
- Learning is feedback enabled for each classifier. i.e., each classifier gets hints on which modes to focus its attention on. First-layer classifiers can learn to output meaningful attributes in favor of the target task (Equation 11 in Section 4.1).
- The inference in our model is fast and its complexity is no more than constant times the complexity of inference in the original classifiers (Section 4.2).
- Scales with large number of tasks. i.e., the same model is applicable if we increase the number from 6 (in this paper) to say, 20. Our model can automatically decide how to compose the tasks sparsely (Section 4.3).
- Ability to handle heterogeneous datasets, i.e., each task could come with its separate labeled data. (Details will be given in Section 5.) This is a very powerful advantage because most vision datasets only have ground truth for one specific task.

2. 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.

5 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. We show our formulation for the general case, where we use Γ_i as the dataset that has labels for i^{th} task. Equation 3, when explicitly decomposed into disjoint datasets, becomes:

$$\log \prod_{i=1}^{n} \prod_{X \in \Gamma_i} P(Y_1, \ldots, Y_n | X; \Theta, \Omega) \tag{18}$$

When data in dataset Γ_i only have labels for the i^{th} task, Equation 18 reduces to maximizing the terms below,

$$\log \prod_{i=1}^{n} \prod_{X \in \Gamma_i} P(Y_i | X, \Theta, \Omega) \tag{19}$$

$$= \sum_{i=1}^{n} \sum_{X \in \Gamma_i} \log \sum_{\mathcal{Z}} P(Y_i | \Psi_i(X), \mathcal{Z}; \omega_i) \prod_{j=1}^{n} P(Z_j | \Psi_j(X); \theta_j) \tag{20}$$

To adjust the importance of data in different datasets, we add turning parameters λ_i's to Equation 20. The selection of λ_i's will be discussed later in this section.

$$\sum_{i=1}^{n} \lambda_i \sum_{X \in \Gamma_i} \log \sum_{\mathcal{Z}} P(Y_i | \Psi_i(X), \mathcal{Z}; \omega_i) \prod_{j=1}^{n} P(Z_j | \Psi_j(X); \theta_j) \tag{21}$$

The corresponding modifications for the feed-forward step (Equation 7 and 8) and the feed-back step (Equation 11) in Section 4 are shown as follows.

Feed-forward Step:

$$\underset{\theta_i}{\text{maximize}} \sum_{j} \lambda_j \sum_{X \in \Gamma_j} \log P(Z_i | \Psi_i(X); \theta_i) \tag{22}$$

$$\underset{\omega_j}{\text{maximize}} \sum_{X \in \Gamma_j} \log P(Y_j | \Psi_j(X), \mathcal{Z}; \omega_j) \tag{23}$$

Feed-back Step: for $X \in \Gamma_j$,

$$\underset{\mathcal{Z}}{\text{maximize}} \log P(Y_j, \mathcal{Z} | X; \theta_1, \ldots, \theta_n, \omega_1, \ldots, \omega_n)$$

$$\Leftrightarrow \underset{\mathcal{Z}}{\text{maximize}} \sum_{i=1}^{n} \log P(Z_i | \Psi_i(X); \theta_i) + \log P(Y_j | \Psi_j(X), \mathcal{Z}; \omega_j) \tag{24}$$

The selection of λ_i results in three 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, λ_i is set to be inversely proportional to the amount of data in the dataset of the i^{th} task. Therefore, the unified FECCM balances the amount of data in different datasets. According to Equation 22, data from different datasets would affect the learning of the first layer classifiers equally. Thus the outputs of the first-layer classifiers tend to equally benefit different tasks on the second layer.
- **One-goal FECCM**: In this instantiation, we set $\lambda_i = 1$ when $i = j$, and $\lambda_i = 0$ when $i \neq j$. This is an extreme setting to favor the specific task j. In this case, the retraining of the first-layer classifiers will only use the feedback from the $Classifier_j$ on the second layer, i.e., only use the dataset with labels for the j^{th} task. Therefore, FECCM degrades to a model with only one target task (the j^{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 j^{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 7).
- **Target-Specific FECCM**: The goal of 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 λ_i for different datasets. In detail, λ_i 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 retraining of the first-layer classifiers can still use data from different datasets (i.e. with different task labels), and thus can better handle the overfitting problem.

6 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 7.

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 is 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,

$$\Phi_j = [\Psi_j \ Z_1 \ Z_2 \ Z_3 \ Z_4 \ Z_5 \ Z_6] \tag{25}$$

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 passed as 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. [48]: 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 [48]. We use an RBF-Kernel SVM classifier [49], as the first-layer scene classifier, and a multi-class logistic classifier for the second layer.

The feature inputs for the first-layer scene classifier $\Psi_1 \in \mathbb{R}^{512}$ is the GIST feature vector for each image [48].

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 specific 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 seek to estimate the depth of every pixel in an image. We evaluate the performance of the estimation 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 [50]. We use a linear regression for the first-level and second-level instantiation of the depth estimation module.

The feature inputs for the first-layer depth estimation $\Psi_2 \in \mathbb{R}^{104}$ are superpixel-level features which capture texture, color and gradient features of the superpixel. This is obtained by convolving the image with Laws' masks and computing the energy and Kurtosis over the superpixel along with the superpixel shape features as described by Saxena et. al. [50].

The output of the first-layer depth estimation $Z_2 \in \mathbb{R}_+$ is the predicted depth of each superpixel 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 superpixels in the image; for the other tasks, we use the 1-dimensional predicted depth for the superpixel/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. [35]: 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. We use a multi-class logistic classifier on each layer for the event classification task.

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 [51], the 12-dimension global color features (mean and variance of RGB and YCrCb color channels over the entire image), and a bias term.

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 specific 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. [52] for our experiments. We use a logistic model as the first-level and second-level classifiers to help estimate the saliency score of every point in the image.

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. [52] and a bias term.

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/superpixel/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 [53] for our experiments. Our object detection module builds on the part-based detector of Felzenszwalb et. al. [54]. We first generate 5 to 100 candidate windows for each image by applying the part-based detector with a low threshold (over-detection). 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 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 [55] plus the detection score from the part-based detector [54]. K depends on the number of scales to be considered and the size of the object template.

The output of a first-layer object detection classifier $Z_5^{(j)} \in \mathbb{R}$ is the estimated "+1" or "0" label for a candidate window to be the j^{th} object. 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/superpixel/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. [56]. We use the dataset and the algorithm by [56] as the first-layer geometric labeling module. In order to reduce the computational time, we avoid the multiple segmentation

and instead use a single segmentation with 100 segments per image. We use a logistic model as the second-layer classifier. 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 superpixel-level features as described by Hoiem et. al. [56].

The output of the first-layer geometry labeling classifier $Z_6 \in \mathbb{R}^3$ is 3-dimensional vector with each element representing the log-odd score for the corresponding superpixel to belong to a geometric category. In order to feed the first-layer geometric labeling output to the second-layer classifiers, for the scene categorization and event categorization tasks, 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 score for the corresponding pixel/superpixel/bounding-box.

7 Experiments and Results

7.1 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 machine learning domains.

We evaluate our proposed method on combining six tasks introduced in Section 6. 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 in Table 1,

- Base model: Our implementation (Section 6) of the algorithm for the 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 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 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.
- FE-CCM (target specific): In this case, we train a specific FE-CCM for each sub-task, by using cross-validation to estimate λ_i's in Equation 21. Different values for λ_i'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 other sub-tasks.

7.2 Datasets

The datasets used are mentioned in Section 6, 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 [53] dataset (1277 train and 2686 test images respectively).

7.3 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 6. 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 [35, 50] or by using adhoc methods to implicitly use information from other tasks [56]. 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 λ_i'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 FECCM 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 FECCM, 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

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 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 [35]	16.7 (MRF) [3] Saxena [50]	83.8 Torralba [49]	82.5 (±0.2) Achanta [52]	88.1 Hoiem [56]	61.5	36.3	39.2	40.7	44.4
						Felzenswalb et. al. [27] (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**

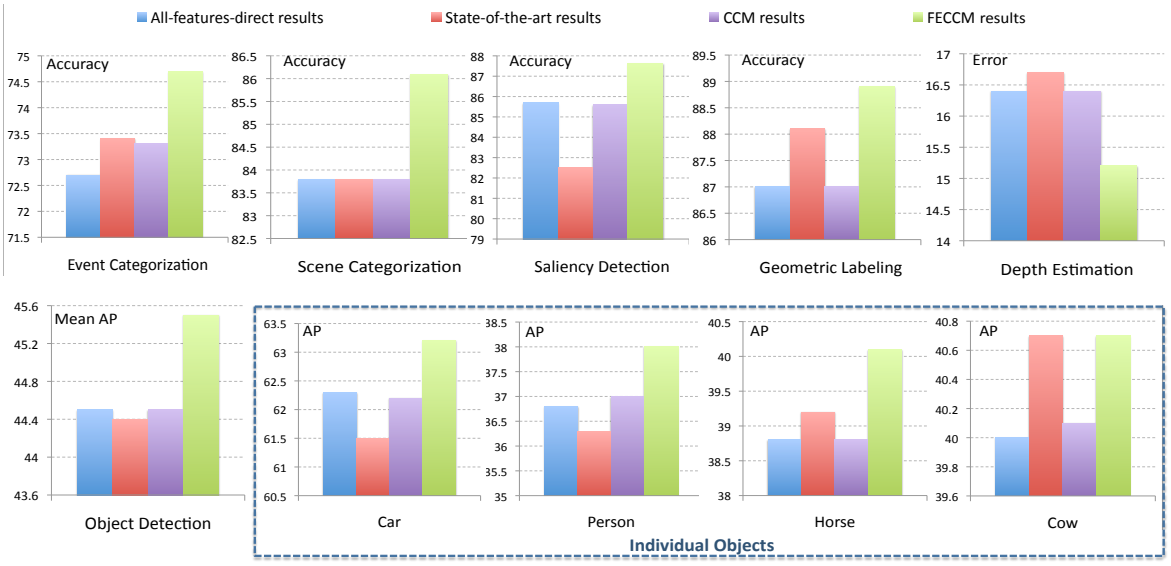


Fig. 3. Results for the six tasks in scene understanding. Figures on the first row give results for: event categorization, scene categorization, saliency detection, geometric labeling, and depth estimation. Figures on the second row give the average results for object detection and the results for the detection of individual object categories: car, person, horse, and cow. Each figure compares three methods: all-features-direct method, state-of-the-art methods, the proposed FECCM method.

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.

7.4 Discussion

FE-CCM allows each classifier in the second layer to learn which information from the other first-layer sub-tasks is useful in the form of weights (in contrast to manually using the information shared across sub-tasks in some prior works). We provide a visualization of the weights for the 6 vision tasks in Figure 6-left. We see that the model agrees with our intuitions that high weights are assigned to the outputs of the same task from the first layer classifier (see high 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 6-right provides a closer look to the positive weights given to the various outputs for a second-level geometric classifier. We observe that high positive weights are assigned to "mountain", "forest", "tall building", etc. for

3. The state-of-the-art method for depth estimation in [50] follows a slightly different testing procedure. In that case, our target-specific FE-CCM method achieves $RMSE = 15.3$.

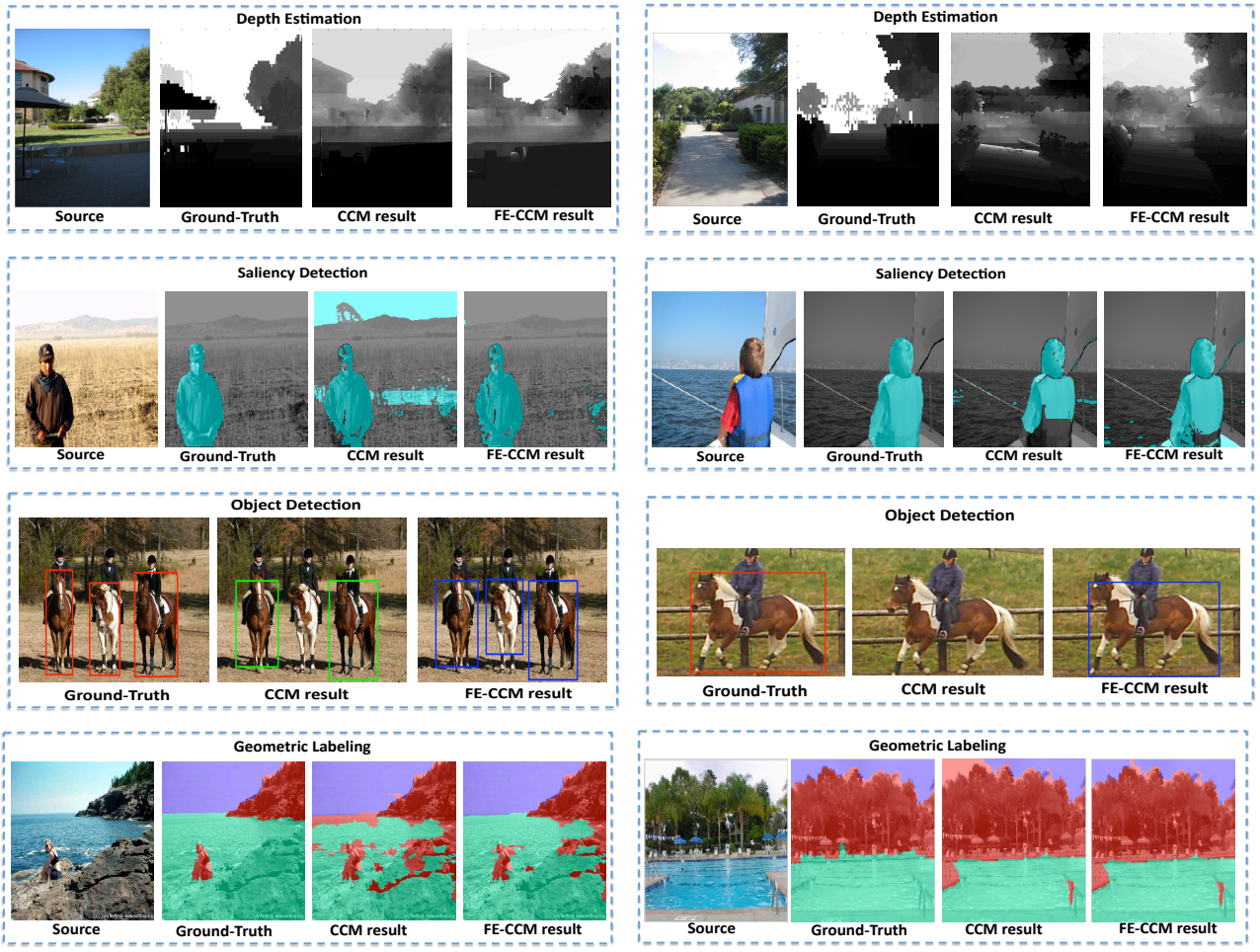
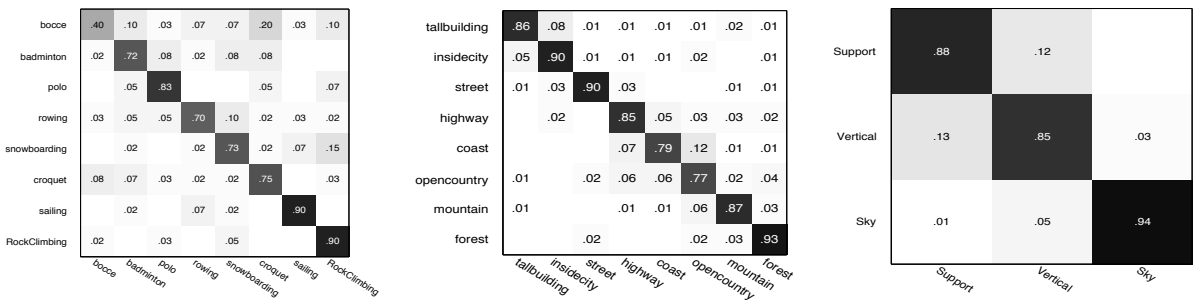


Fig. 4. Results showing improvement using the proposed model. From top to bottom: Depth estimation, Saliency detection, Object detection, Geometric labeling. All depth maps in depth estimation are at the same scale (black means near and white means far); Salient region in saliency detection are indicated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



(a) (b) (c)

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.

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 7a visualizes the weights given to the depth attributes (first-layer depth outputs) for the task of event categorization. Figure 7b shows the same for the task of scene categorization. We see that the depth plays an important role in these tasks. In Figure 7a, 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

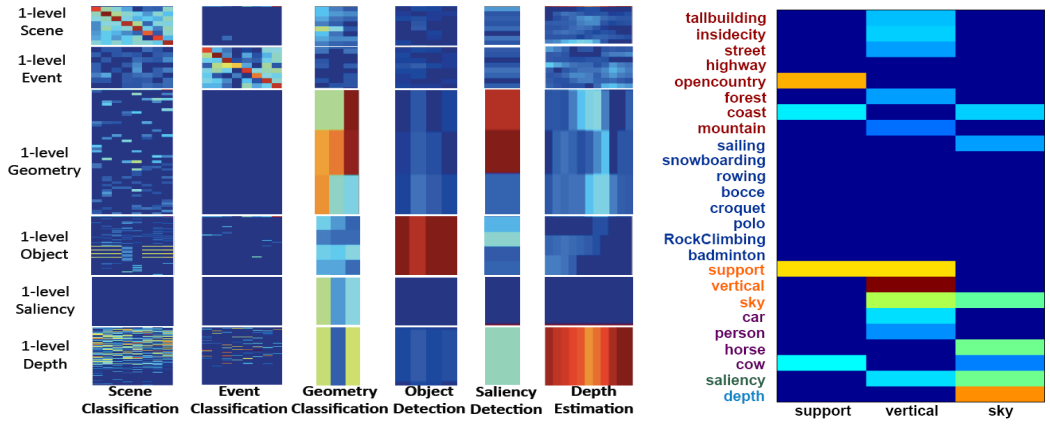
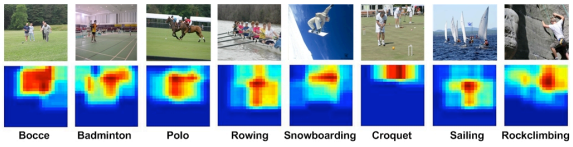
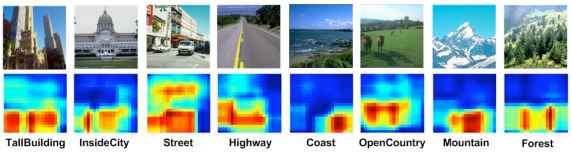


Fig. 6. (Left) The *absolute* values of the weight vectors for second-level classifiers, i.e. ω. Each column shows the contribution of the various tasks towards a certain task. (Right) Detailed illustration of the *positive* values in the weight vector for a second-level geometric classifier. (Note: Blue is low and Red is high)



(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. 7. 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.

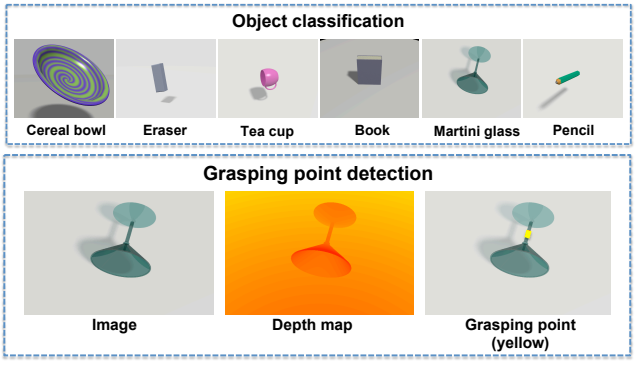


Fig. 8. Examples in the dataset used for the grasping robot experiments. The two tasks considered were a six-class, object classification task and grasping point detection task (Best viewed in color).

had automatically figured this out (see Figure 7b).

Stability of FECCM: In this paper, we have presented results for six sub-tasks. In order to find out how our method scales with different combination 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. We then combined four tasks: event categorization, scene categorization, depth estimation, and saliency detection, and got improvements in all these sub-tasks [57]. Finally, we also combined scene categorization, geometric labeling, and object detection, in order to build a one-goal FE-CCM for an object detector used in a shoe finding robot, and the performance improvement was similar. A video of our robot fetching the shoe is available at [58].

8 Scene Understanding for Robotic Applications

In order to show the applicability of our FECCM to problems across different scene understanding domains, we also used the proposed method in multiple robotic applications.

8.1 Robotic Grasping

Given an image and a depthmap (Figure 8), the goal of the learning algorithm in a grasping robot is to select a point at which to grasp the object (this location is called the grasp point, [59]). It turns out that different categories of objects demand different strategies for grasping. In prior work, Saxena et al. [59] 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. [59] which spans 6 object categories and also includes an aligned pixel level depth map for each image, as shown in Figure 8. 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 open-source code given by [59]). 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

TABLE 2
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**

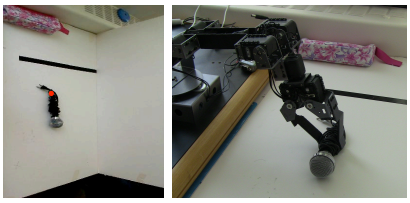


Fig. 9. Left: the grasping point detected by our algorithm. Right: Our robot grasping an object using our algorithm.

regression over the features. The output of the regression 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.

Results: We evaluate our algorithm on a dataset published in [59], 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 2 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 9 shows an example of our robot grasping an object using our algorithm.

8.2 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 FECCM 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 [60] and classify an image into one of the four categories: bedroom, living room, kitchen and office. We use the same features and classifiers for scene categorization as in Section 6.

Fig. 10. Left: the shoe-finding robot, which has a camera to take photos of a scene. Right: the shoed detected using our algorithm.

For geometric labeling, we use the Indoor Layout Data [61] and assign each pixel to one of three geometry classes: ground, wall and ceiling. We use the same features and the same pixel-based geometry classifiers as in Section 6.

For object detection, we use the PASCAL 2007 Dataset [62] 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 [27] to create candidate windows, and then use the same classifiers as described in Section 6.

Results: We use this method to build a shoe-finding robot, as shown on the left Figure 10. 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, by using our FECCM model, the robot learns to take advantage of the other tasks and performs a more robust shoe detection. One example is given on the right of Figure 10. For more details and videos, please see [58].

9 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 Industrial Technology Research Institute in Taiwan and Kodak for their financial support in this research. We thank Anish Nahar, Matthew Cong, TP Wong, Norris Xu, and Colin Ponce for help with the robotic experiments. We also thank John Platt and Daphne Koller for useful discussions.

REFERENCES

[1] D. Hoiem, A. A. Efros, and M. Hebert, "Closing the loop on scene interpretation," in *CVPR*, 2008.

[2] L.-J. Li, R. Socher, and L. Fei-Fei, "Towards total scene understanding: Classification, annotation and segmentation in an automatic framework," in *CVPR*, 2009.

[3] 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.

[4] S. Kumar and M. Hebert, "A hierarchical field framework for unified context-based classification," in *ICCV*, 2005.

[5] C. Sutton and A. McCallum, "Joint parsing and semantic role labeling," in *CoNLL*, 2005.

[6] D. Parikh, C. Zitnick, and T. Chen, "From appearance to context-based recognition: Dense labeling in small images," *CVPR*, 2008.

[7] A. Toshev, B. Taskar, and K. Daniilidis, "Object detection via boundary structure segmentation," in *CVPR*, 2010.

[8] A. Agarwal and B. Triggs, "Monocular human motion capture with a mixture of regressors," in *IEEE Workshop Vision for HCI, CVPR*, 2005.

[9] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3d scene structure from a single still image," *IEEE PAMI*, vol. 30, no. 5, 2009.

[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] Z. Tu, "Auto-context and its application to high-level vision tasks," in *CVPR*, 2008.

[19] J. Kittler, M. Hatef, R. P. Duin, and J. Matas, "On combining classifiers," *PAMI*, vol. 20, pp. 226–239, 1998.

[20] A. Torralba, "Contextual priming for object detection," *Int. J. Comput. Vision*, vol. 53, no. 2, pp. 169–191, 2003.

[21] A. Torralba and A. Oliva, "Depth estimation from image structure," *IEEE PAMI*, vol. 24, pp. 1226–1238, September 2002.

[22] D. Hoiem, A. A. Efros, and M. Hebert, "Putting objects in perspective," in *CVPR*, 2006.

[23] D. Park, D. Ramanan, and C. Fowlkes, "Multiresolution models for object detection," in *ECCV*, 2010.

[24] G. Heitz and D. Koller, "Learning spatial context: Using stuff to find things," in *ECCV*, 2008.

[25] J. Lim, P. Arbel andez, C. Gu, and J. Malik, "Context by region ancestry," in *ICCV*, 2009.

[26] S. Kumar and M. Hebert, "A hierarchical field framework for unified context-based classification," in *ICCV*, 2005.

[27] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *PAMI*, 2009.

[28] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie, "Objects in context," in *ICCV*, 2007.

[29] D. Parikh, C. Zitnick, and T. Chen, "From appearance to context-based recognition: Dense labeling in small images," in *CVPR*, 2008.

[30] C. Desai, D. Ramanan, and C. Fowlkes, "Discriminative models for multi-class object layout," in *ICCV*, 2009.

[31] B. Yao and L. Fei-Fei, "Modeling mutual context of object and human pose in human-object interaction activities," in *CVPR*, 2010.

[32] C. Galleguillos, B. McFee, S. Belongie, and G. Lanckriet, "Multi-class object localization by combining local contextual interactions," in *CVPR*, 2010.

[33] S. Divvala, D. Hoiem, J. Hays, A. Efros, and M. Hebert, "An empirical study of context in object detection," in *CVPR*, 2009.

[34] M. Blaschko and C. Lampert, "Object localization with global and local context kernels," in *BMVC*, 2009.

[35] L. Li and L. Fei-Fei, "What, where and who? classifying event by scene and object recognition," in *ICCV*, 2007.

[36] Y. Bengio and Y. LeCun, "Scaling learning algorithms towards ai," in *Large-Scale Kernel Machines*, 2007.

[37] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, pp. 41–75, 1997.

[38] 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.

[39] I. Goodfellow, Q. Le, A. Saxena, H. Lee, and A. Ng, "Measuring invariances in deep networks," in *NIPS*, 2009.

[40] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," in *N. Comp*, 2006.

[41] M. Zeiler, D. Krishnan, G. Taylor, and R. Fergus, "Deconvolutional networks," in *CVPR*, 2010.

[42] I. Tsochantaridis, T. Hofmann, and T. Joachims, "Support vector machine learning for interdependent and structured output spaces," in *ICML*, 2004.

[43] B. Taskar, C. Guestrin, and D. Koller, "Max-margin markov networks," in *NIPS*, 2003.

[44] 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.

[45] 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.

[46] M. Gibbs and D. Mackay, "Variational gaussian process classifiers," *Neural Networks, IEEE Trans*, 2000.

[47] 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.

[48] 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.

[49] A. Torralba and A. Oliva, "MIT outdoor scene dataset," http://people.csail.mit.edu/torralba/code/spatialenvelope/index.html.

[50] A. Saxena, S. H. Chung, and A. Y. Ng, "3-d depth reconstruction from a single still image," *IJCV*, vol. 76, 2007.

[51] 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.

[52] R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk, "Frequency-tuned Salient Region Detection," in *CVPR*, 2009.

[53] 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

[54] 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/.

[55] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *CVPR*, 2005.

[56] D. Hoiem, A. A. Efros, and M. Hebert, "Putting objects in perspective," *IJCV*, 2008.

[57] A. Kowdle, C. Li, A. Saxena, and T. Chen, "A generic model to compose vision modules for holistic scene understanding," in *Workshop on Parts and Attributes, ECCV*, 2010.

[58] C. Li, T. Wong, N. Xu, and A. Saxena, "FECCM for scene unverstanding: Helping the robot to learn multiple tasks," in *Video track ICRA 2011. Online URL: http://www.cs.cornell.edu/ asaxena/ccm/*, 2010.

[59] A. Saxena, J. Driemeyer, J. Kearns, and A. Y. Ng, "Robotic grasping of novel objects," in *NIPS*, 2006.

[60] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," *CVPR*, 2005.

[61] V. Hedau, D. Hoiem, and D. Forsyth, "Recovering the spatial layout of cluttered rooms," in *ICCV*, 2009.

[62] 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.

Congcong Li received the B.E. and M.S. degrees from the Department of Electronic Engineering, Tsinghua University, Beijing, China, respectively, in 2005 and 2007. She started her Ph.D. study in the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, from 2007 and moved to Cornell University, Ithaca, NY, in 2009. She is currently pursuing the Ph.D. degree in the School of Electrical and Computer Engineering, Cornell University. Her research interests include computer vision, machine learning, and image analysis.

Adarsh Kowdle received his B.E. degree from the Department of Electronics and Communication, Rashtreeya Vidyalaya College of Engineering (Bangalore, India), in 2007. He worked with Ittiam Systems (Bangalore, India) from June 2007 to July 2008. He started his Ph.D. study in the Department of Electrical and Computer Engineering, Cornell University (Ithaca, NY), in 2009, before which he spent a semester at the Department of Electrical and Computer Engineering, Carnegie Mellon University (Pittsburgh, PA). He is currently pursuing the Ph.D. degree in the School of Electrical and Computer Engineering, Cornell University. His research interests include computer vision, machine learning, and human-computer interaction.

Ashutosh Saxena 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. He has worked in Bose Corporation, CSIRO (Australia) and Microsoft in the past, and has founded Zunavision in 2008.

Ashutosh 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 (such as STAIR) 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.