Toward 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 subtasks, such as scene categorization, depth estimation, object detection, etc. Each of these subtasks 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 subtasks while requiring only a “black box” interface to the original classifier for each subtask. 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 subtasks 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 subtasks, such as depth estimation, scene categorization, saliency detection, object detection, event categorization, etc. (See Fig. 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 subtasks. Several independent efforts have resulted in good classifiers for each of these subtasks. In practice, we see that the subtasks 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 subtasks, allowing them to share information and guide the classifiers toward 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], [2], [3], [4], [5], [6], [7], [8], [9], [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 subtasks. However, these carefully engineered models are often tricky to modify, or even to simply reimplement 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 four of the 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 information to earlier stages 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 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 data sets, i.e., our model does not require a data point to have labels for all the subtasks, and hence it scales well with heterogeneous data sets.

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 classifiers 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 six subtasks 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 data sets 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 Fig. 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 is it? 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 subtask. If we are considering depth estimation (see Fig. 1), then the label would be \( Y_i \in S_1 = \mathbb{R}_+^{100 \times 100} \) for continuous values of depth in a \( 100 \times 100 \) output. For scene categorization, we will have \( Y_i \in S_2 = \{1, \ldots, K\} \) for \( K \) scene classes. If we have \( n \) subtasks, 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 want to solve different combinations of the subtasks depending on the situation. The goal of this work is to design an algorithm that does not depend on the particular subtasks in question.

2.2 Related Work

2.2.1 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 multilevel perceptrons where the output of the first layer of perceptrons is passed on as input to the next layer [12], [13], [14]. However, it is often hard to train neural networks and gain an insight into their operation, making it hard to use 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.

2.2.2 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 are 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.

2.2.3 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 multiclass segmentation and object detection. Li et al. [24] modeled multiple 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], [27], [28]) can also be a viable option for the setting of combining multiple tasks. There has been a 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 multitask settings with disjoint data sets. 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], [8], [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.

2.2.4 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 and Oliva [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 a contextual information for pedestrian detection. Many works also model context to capture the local interactions between neighboring regions [35], [36], [37], objects [38], [39], [40], [41], [42], or both [43], [44], [45]. These methods improve the performance of some specific tasks by combining information from different aspects. However, most of these methods cannot be applied to cases when we only have “black box” classifiers for the individual tasks.

2.2.5 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.

2.2.6 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 multilayer 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], [51], [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 multilayered architecture does not suffer from local minima during optimization. At a high level, we can also look at our work as a multilayered 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 subtasks.

3 Feedback Enabled Cascaded Classification Models

In the field of scene understanding, a lot of independent research into each of the vision subtasks has led to excellent classifiers. These independent classifiers are typically trained on different or heterogenous data sets due to the lack of ground-truth labels for all the subtasks. In addition, each of these classifiers comes with its 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 data set $X$, features extracted $\Psi(X)$ and the target outputs of the $i$th task $Y_i$, the black-box classifier has some internal learning function $f_{\text{learn}}^i$ with parameters $\theta_i$ that optimizes the mapping from the inputs to the outputs for the training data.$^1$ Once the parameters have been learned, given a new data point, $X$ with features

---

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., $X$, $Y_i$, etc.) are used for a data set.
The proposed feedback enabled cascaded classification model for combining related classifiers. \(\Psi_i(X)\) = Features corresponding to Classifier, extracted from image \(X\). \(Z_i\) = Output of the Classifier, in the first stage parameterized by \(\theta_i\). \(Y_i\) = Output of the Classifier, in the second stage parameterized by \(\omega_i\).

The inferred output for each classifier is given and our goal is to infer the final outputs \(Y_i\). Using the learned parameters \(\theta_i\) for the first level of classifiers and \(\omega_i\) 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:

\[
\hat{Z}_i = \text{optimize} \ f_{\text{infer}}^i(\Psi_i(X), Z_i; \theta_i), \tag{3}
\]

\[
\hat{Y}_i = \text{optimize} \ f_{\text{infer}}^i(\hat{Z}_i, Y_i; \omega_i). \tag{4}
\]

The inference algorithm is given in Algorithm 1. This method allows us to use the internal inference function (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 classifiers.

**Algorithm 1. Inference**

1. **Inference for first layer:**
   
   for \(i = 1 : n\)
   
   Infer the outputs of the \(i^{th}\) classifier using (3);
   
   end

2. **Inference for second layer:**
   
   for \(i = 1 : n\)
   
   Infer the outputs of the \(i^{th}\) classifier using (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 $Z$ (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 $Z$ 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. ground truth), 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 Fig. 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 (Feedback 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 $Z$ with the ground-truth $Y$.

2. Do until convergence or maximum iteration: 

   **Feed-forward step:** Fix latent variables $Z$, estimate the parameters $\Theta$ and $\Omega$ using (5) and (6).

   **Feedback step:** Fix the parameters $\Theta$ and $\Omega$, compute latent variables $Z$ using (7).

**Initialization.** We initialize the model by setting the latent variables $Z_i$s to the ground truth, 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 classifiers 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 $\Theta$ and $\Omega$. 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 $\theta_i$ is precisely the learning problem of the “black-box classifier,” and learning $\omega_i$ 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 (1):

$$\hat{\theta}_i = \text{optimize } f_{\text{learn}}(\Psi_i(X), Z_i; \hat{\theta}_i),$$

$$\hat{\omega}_i = \text{optimize } f_{\text{learn}}^h(\Psi_i(X; Z_I), Y_i; \hat{\omega}_i).$$

We now have the parameters for all the classifiers.

**Feedback 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 feedback 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 $\hat{\theta}_i$s and $\hat{\omega}_i$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 feedback step:

$$\text{optimize } \sum_{Z_i} (J_1(\Psi_i(X), Z_i; \hat{\theta}_i) + J_2(\Psi_i(X), Z, Y_i; \hat{\omega}_i)), \quad (7)$$

where $J_1$ and $J_2$ are functions, respectively, related to the first-layer classifiers and the second-layer classifiers. one option is to have $J_1(\Psi_i(X), Z_i; \hat{\theta}_i) = f_{\text{infer}}(\Psi_i(X), Z_i; \hat{\theta}_i)$ and $J_2(\Psi_i(X), Z, Y_i; \hat{\omega}_i) = f_{\text{infer}}^h(\Psi_i(X), Z, Y_i; \hat{\omega}_i)$ 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 relearn 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., $\log P(Y|X)$, where $X$ is an image belonging to training set $\Gamma$. Therefore, the goal of the proposed model shown in Fig. 2 is to maximize

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

To introduce the hidden variables $Z_i$s, we expand (8) as follows, using the independencies represented by the directed model in Fig. 2:

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

$$= \sum_{X \in \Gamma} \log \sum_{Z} \prod_{i=1}^n P(Y_i|\Psi_i(X), Z_i; \hat{\theta}_i) P(Z_i|\Psi_i(X); \hat{\theta}_i). \quad (10)$$

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 in 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

$$\text{maximize } \sum_{\theta_1, \ldots, \theta_n, \omega_1, \ldots, \omega_n} \prod_{X \in \Gamma} P(Y_i|\Psi_i(X), Z_i; \hat{\theta}_i) P(Z_i|\Psi_i(X); \hat{\theta}_i). \quad (11)$$

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
Note that the optimization problem nicely breaks down into the subproblems of training the individual classifier for the respective subtasks. We can solve each subproblem 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 ((5) and (6)).

**Feedback step.** In this step, we estimate the values of the latent variables \(Z_i\) 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 (10), we get the following optimization problem:

\[
\begin{align*}
\max_{Z} \sum_{i=1}^{n} \left( \log P(Y_i | \Psi_i(X), Z; \hat{\omega}_i) + \log P(Z_i | \Psi_i(X); \hat{\theta}_i) \right) \equiv \\
\max_{Z} \log P(Y_1, \ldots, Y_n, Z | \hat{\theta}_1, \ldots, \hat{\theta}_n, \hat{\omega}_1, \ldots, \hat{\omega}_n) \Rightarrow \\
\max_{Z} \sum_{i=1}^{n} \log P(Y_i | \Psi_i(X), Z; \hat{\omega}_i) + \log P(Z_i | \Psi_i(X); \hat{\theta}_i)
\end{align*}
\]

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 (14) is same as (7) with \(J_1^i(\Psi_i(X), Z_i; \hat{\theta}_i) = \log P(Z_i | \Psi_i(X); \hat{\theta}_i)\) and \(J_2^i(\Psi_i(X), Z_i, Y_i; \hat{\omega}_i) = \log P(Y_i | \Psi_i(X), Z_i; \hat{\omega}_i)\).

In some cases, the classifier log-likelihoods in (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:

\[
J_1^i(\Psi_i(X), Z_i; \hat{\theta}_i) = \| Z_i - \hat{Z}_i \|^2 \\
\text{s.t. } \hat{Z}_i = \text{optimize } f_{\text{infer}}^i(\Psi_i(X), Z_i; \hat{\theta}_i).
\]

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

\[
\min_{\alpha_i} \sum_{X \in \mathcal{T}} \| \hat{Y}_i - \alpha_i^T [\Psi_i(X), \hat{Z}] \|^2_2,
\]

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

\[
J_2^i(\Psi_i(X), Z_i, Y_i; \hat{\omega}_i) = \| Y_i - \alpha_i^T [\Psi_i(X), Z] \|^2_2.
\]

**Sparsity.** Note that the parameter \(\alpha_i\) 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 nonzero weights, i.e., only a few nonzero entries in \(\alpha_i\). We do this by introducing the \(l_1\) sparsity in the parameters [56]. So, (16) is extended as follows:

\[
\min_{\alpha_i} \sum_{X \in \mathcal{T}} \| \hat{Y}_i - \alpha_i^T [\Psi_i(X), \hat{Z}] \|^2_2 + \beta|\alpha_i|.
\]

**Inference.** As introduced in Section 3.2, our inference procedure consists of two steps: First maximize over hidden variable \(Z\) and then maximize over \(Y\):

\[
\hat{Z} = \text{argmax}_Z \log P(Z | X, \hat{\theta}), \quad \hat{Y} = \text{argmax}_Y \log P(Y | \hat{Z}, X, \hat{\Theta}).
\]

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

\[
\hat{Z}_i = \text{argmax}_Z \log P(Z_i | \Psi_i(X); \hat{\theta}_i), \quad i = 1, \ldots, n, \quad (21)
\]

\[
\hat{Y}_i = \text{argmax}_Y \log P(Y_i | \Psi_i(X); \hat{Z}; \hat{\omega}_i), \quad i = 1, \ldots, n. \quad (22)
\]

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

### 4 Training with Heterogeneous Data Sets

Often real data sets are disjoint for different tasks, i.e., each data point 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 \(\Gamma_i\) as the data set that has labels only for the \(i\)th task.

In the following, we provide the modifications to the feed-forward step and the feedback step while dealing with disjoint data sets, i.e., data in data set \(\Gamma_i\) 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 data sets in order to relearn 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

\[
3. \text{Another alternative would have been to maximize } P(Y | X) = \sum_{Z} P(Y, Z | X); \text{ however, this would require marginalization over the variable } Z \text{ which is expensive to compute.}
\]
Since a data point in the set \(J_i\) only has ground-truth label for the corresponding task: Toward this goal, \(\pi_j\) is set to be inversely proportional to the amount of data in the data set of the \(j\)th task. Therefore, the unified FECCM balances the amount of data in different data sets, based on \(23\).

- **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 \(\text{Classifier}_k\) on the second layer, i.e., only use the data set 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 \(\theta_i/s\) and \(\omega_i/s\), 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 \(\pi_j\) for different data sets. In detail, \(\pi_j\) 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 retraining of the first-layer classifiers can still use data from different data sets (i.e., with different task labels). Here, we train different models, i.e., different \(\theta_i/s\) and \(\omega_i/s\), for different target tasks.

### 4.1 FECCM: Different Instantiations

The parameters \(\pi_i\) 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 data sets (i.e., with different task labels). Toward this goal, \(\pi_j\) is set to be inversely proportional to the amount of data in the data set of the \(j\)th task. Therefore, the unified FECCM balances the amount of data in different data sets, based on \(23\).

| Model                          | Event Categorization (% Accuracy) | Depth Estimation (RMSE in m) | Scene Categorization (% Accuracy) | Saliency Detection (% Accuracy) | Geometric Labeling (% Accuracy) | Cat | Person | Horse | Cow | Mean |
|-------------------------------|---------------------------------|------------------------------|----------------------------------|-------------------------------|---------------------------------|-----|--------|-------|-----|------|
| Images in testset             | 1579                            | 400                          | 2688                             | 1000                          | 300                             |     |        |       |     |      |
| Chance                        | 22.5                            | 24.6                         | 22.5                             | 50                            | 33.3                            |     |        |       |     |      |
| Our base model               | 71.8 \(\pm 0.8\)               | 16.7 \(\pm 0.4\)            | 83.8 \(\pm 0.2\)                | 85.2 \(\pm 0.2\)             | 86.2 \(\pm 0.2\)               | 62.4| 36.3   | 39.0  | 39.9| 44.4 |
| All-features-direct          | 72.7 \(\pm 0.8\)               | 16.4 \(\pm 0.4\)            | 83.8 \(\pm 0.4\)                | 85.7 \(\pm 0.2\)             | 87.0 \(\pm 0.6\)               | 62.3| 36.8   | 38.8  | 40.0| 44.5 |
| State-of-the-art model       | 73.6                            | 16.7 \(\text{MRF}^4\)       | 83.8                            | 82.5 \(\pm 0.2\)             | 88.1                            | 61.5| 36.3   | 39.2  | 40.7| 44.4 |
| (reported)                  | [14]                            |                              | [59]                             |                               | [58]                            |     |        |       |     |      |
| CCM [11] (our implementation)| 73.3 \(\pm 1.6\)               | 16.4 \(\pm 0.4\)            | 83.8 \(\pm 0.6\)                | 85.6 \(\pm 0.2\)             | 87.0 \(\pm 0.6\)               | 62.2| 37.0   | 38.8  | 40.1| 44.5 |
| PE-CCM (uniform)            | 74.3 \(\pm 0.6\)               | 15.5 \(\pm 0.2\)            | 85.9 \(\pm 0.3\)                | 86.2 \(\pm 0.2\)             | 88.6 \(\pm 0.2\)               | 63.2| 37.6   | 40.1  | 40.5| 45.4 |
| PE-CCM (one goal)           | 74.2 \(\pm 0.8\)               | 15.3 \(\pm 0.4\)            | 85.8 \(\pm 0.5\)                | 87.1 \(\pm 0.2\)             | 88.6 \(\pm 0.3\)               | 63.2| 37.9   | 40.1  | 40.7| 45.5 |
| PE-CCM (target specific)    | 74.7 \(\pm 0.6\)               | 15.2 \(\pm 0.2\)            | 86.1 \(\pm 0.2\)                | 87.6 \(\pm 0.2\)             | 88.9 \(\pm 0.2\)               | 63.2| 38.0   | 40.1  | 40.7| 45.5 |

**Summary of Results for the SIX Vision Tasks**

*Note: Bold face corresponds to our model performing equally as well or better than state of the art.*

Our method improves performance in every single task.

**Feedback step.** In this step, we change \(7\) as follows: Since a data point in the set \(\Gamma_j\) only has ground-truth label for the corresponding task: Toward this goal, \(\pi_j\) is set to be inversely proportional to the amount of data in the data set of the \(j\)th task. Therefore, the unified FECCM balances the amount of data in different data sets, based on \(23\).

- **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 \(\text{Classifier}_k\) on the second layer, i.e., only use the data set 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 \(\theta_i/s\) and \(\omega_i/s\), 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 \(\pi_j\) for different data sets. In detail, \(\pi_j\) 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 retraining of the first-layer classifiers can still use data from different data sets (i.e., with different task labels). Here, we train different models, i.e., different \(\theta_i/s\) and \(\omega_i/s\), 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 subtasks are our “base model” shown in Table 1. In some subtasks, our base model will be simpler than the state-of-the-art models (that
are often hand-tuned for the specific subtasks, 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 subtasks (on the same standard respective data sets) 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 six 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 \( z_j \). At the second layer, in addition to the original features \( z_j \), the inputs include the outputs from the first layer classifiers. This is given by

\[
\Phi_j = [\psi_1 z_1 z_2 z_3 z_4 z_5 z_6],
\]

where \( \Phi_j \) is the input feature vector for the \( j \)th task on the second layer and \( z_j (i = 1, \ldots, 6) \) 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 eight categories defined by Oliva and Torralba [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 data set [57]. The feature inputs for the first-layer scene classifier \( \psi_1 \in \mathbb{R}^{312} \) is the GIST feature [57], extracted at 4 \( \times \) 4 regions of the image, on four scales and eight orientations.

We use an RBF-Kernel SVM classifier [58] as the first-layer scene classifier, and a multiclass logistic classifier for the second layer. The output of the first-layer scene classifier \( z_1 \in \mathbb{R}^8 \) is an eight-dimensional vector where each element represents the log-odds score of the corresponding image belonging to a scene category. This eight-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 data set [59], [60]. We uniformly divide each image into 55 \( \times \) 305 patches as [59]. The feature inputs for the first-layer depth estimation classifier \( \psi_2 \in \mathbb{R}^{404} \) 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 one-dimensional predicted depth for the patch/pixel/bounding-box, etc.

**Event categorization.** For event categorization, we classify an image into one of the eight sports events as defined by Li and Fei-Fei [46]: bocce, badminton, polo, rowing, snowboarding, croquet, sailing, and rock climbing. For evaluation, we compute the rate of correctly 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 multiclass logistic classifier on each layer for event classification. The output of the first-layer event classifier \( z_3 \in \mathbb{R}^8 \) is an eight-dimensional vector where each element represents the log-odd score of a pixel belonging to an event category. This eight-dimensional output is fed 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 nonsalient. We use the saliency detection data set used by Achanta et al. [62] for our experiments. The feature inputs for the first-layer saliency classifier \( \psi_4 \in \mathbb{R}^4 \) include the three-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 \( z_4 \) 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 one-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 trainset 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 (overdetection). 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 outputs 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 four 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 one-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 \( \Psi \in \mathbb{R}^{12} \) are the region-based features as described by Hoiem et al. [33].

We use the data set and the algorithm by Hoiem et al. [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 \( Z_0 \in \mathbb{R}^3 \) is a three-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 three-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 subtasks. We believe this is a powerful algorithm in that, while independent efforts toward each subtask have led to state-of-the-art algorithms that require intricate modeling for that specific subtask, the proposed approach is a unified model which can beat the state-of-the-art performance in each subtask 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 subtasks in each of the domains, we evaluate our performance on the standard data set for that subtask (and compare against the specifically designed state-of-the-art algorithm for that data set). Note that, with such disjoint yet practical data sets, 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 subtask, which serves as a base model for our FE-CCM. (The base model uses less information than state-of-the-art algorithms for some subtasks.)
- All-features-direct: A classifier that takes all the features of all subtasks, appends them together, and builds a separate classifier for each subtask.
- State-of-the-art model: The state-of-the-art algorithm for each subtask, respectively, on that specific data set.
- CCM: The cascaded classifier model by Heitz et al. [11], which we reimplement for six subtasks.
- FE-CCM (unified): This is our proposed model. Note that this is one single model which maximizes the joint likelihood of all the subtasks.
- FE-CCM (one goal): In this case, we have only one subtask instantiated on the second layer, and the goal is to optimize the outputs of that subtask. We train a specific one-goal FE-CCM for each subtask.
- FE-CCM (target specific): In this case, we train a specific FE-CCM for each subtask, by using cross validation to estimate \( \pi_s \) in (23). Different values for \( \pi_s \) result in different parameters learned for each FE-CCM.

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

6.2 Data Sets

The data sets used are mentioned in Section 5, and the number of test images in each data set is shown in Table 1. For each data set, 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 subtasks to evaluate the performance on each task. We do not do cross validation on an object detection as it is a standard on the PASCAL 2006 [63] data set (1,277 train and 2,686 test images, respectively).

6.3 Results

To quantitatively evaluate our method for each of the subtasks, we consider the metrics appropriate to each of the six tasks in Section 5. Table 1 and Fig. 3 show that FE-CCM not only beats state of the art in all the tasks but it also does so 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 ad hoc 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 \( \pi_s \) to optimize for each task independently. Though the unified FE-CCM achieves slightly worse performance, it jointly optimizes 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, Fig. 3 shows the results for CCM (which is a cascade without feedback information) and all-features-direct (which uses features 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 Fig. 4. In comparison to CCM, FE-CCM leads to a better depth estimation of the sky and the ground, and it leads to better coverage and an accurate labeling of the salient region in the image, and it also leads to better geometric labeling and object detection. Fig. 5 also provides the confusion matrices for the three tasks: scene categorization, event categorization, and geometric labeling.

Fig. 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 data sets. 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 data set 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 data sets 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 data sets outperforms the CCM method trained with fully labeled data sets, 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.

Fig. 7 illustrates the first-layer outputs of a test image, respectively, at initialization and at the fifth iteration. Our 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 is 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 subtasks is useful, and this can be seen in the learned weights for the second-layer. We provide a visualization of the weights for the six vision tasks in Fig. 8a. We see that the model agrees with our intuition 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.

Fig. 8b 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.

Fig. 8c visualizes the weights given to the depth attributes (first-layer depth outputs) for the task of event categorization. Fig. 8d shows the same for the task of scene categorization. We see that the depth plays an important role in these tasks. In Fig. 8c, we observe that most event categories rely on the middle part of the image, where the main objects of the event are often located. For example, 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 Fig. 8d).

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

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 depth map (Fig. 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 data set by Saxena et al. [66] which spans six object categories and also includes an aligned pixel level depth map for each image, as shown in Fig. 9. The six object categories include spherically symmetric objects such as cereal bowl, rectangular objects such as eraser, martini glass, books, cups, and long objects such as pencil.

For grasp point detection, we compute image and depth map features at each point in the image (using codes given by Saxena et al. [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.

Fig. 9. Examples in the data set used for the grasping robot experiments. The two tasks considered were a six-class, object classification task and grasping point detection task.
TABLE 3

Summary of Results for the Robotic Grasping Experiment

| Model          | Grasping point Detection (% accuracy) | Object Classification (% accuracy) |
|----------------|---------------------------------------|-----------------------------------|
| Images in test set | 6000                                 | 1200                              |
| Chance          | 50                                    | 16.7                              |
| All features direct | 87.7                                 | 45.8                              |
| Our base-model  | 87.7                                  | 45.8                              |
| CCM (Heitz et al.) | 95.5                                 | 49.5                              |
| FE-CCM          | 92.2                                  | 49.7                              |

Our method improves performance in every single task.

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 six-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 data set published in [66], and perform cross validation to evaluate the performance on each task. We use 6,000 images for grasping point detection (3,000 for training and 3,000 for testing) and 1,200 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). Fig. 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 a living room are more likely to have objects (e.g., shoes) than other types of scenes such as a 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 Data Set [68] and 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 Data Set [70] and our own shoe data set 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 in Fig. 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. Fig. 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 subtasks) improves performance significantly across all the subtasks 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 toward holistic scene understanding.

**ACKNOWLEDGMENTS**

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

**REFERENCES**

[1] S. Kumar and M. Hebert, “A Hierarchical Field Framework for Unified Context-Based Classification,” Proc. 10th IEEE Int’l Conf. Computer Vision, 2005.

[2] A. Saxena, M. Sun, and A.Y. Ng, “Make3d: Learning 3d Scene Structure from a Single Still Image,” IEEE Trans. Pattern and Machine Intelligence, vol. 30, no. 5, pp. 824-840, May 2009.

[3] D. Hoiem, A.A. Efros, and M. Hebert, “Closing the Loop on Scene Interpretation,” Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2008.
M. Gibbs and D. Mackay, “Variational Gaussian Process Classifiers,” IEEE Trans. Neural Networks, vol. 11, no. 6, pp. 1458-1464, Nov. 2000.

J. Mairal, M. Leordeanu, F. Bach, M. Hebert, and J. Ponce, “Discriminative Sparse Image Models for Class-Specific Edge Detection and Image Interpretation,” Proc. 10th European Conf. Computer Vision, 2008.

A. Oliva and A. Torralba, “Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope,” Int’l J. Computer Vision, vol. 42, pp. 145-175, 2001.

A. Torralba and A. Oliva, “MIT Outdoor Scene Data Set,” http://people.csail.mit.edu/torralba/code/spatialenvelope/index.html, 2012.

A. Saxena, S.H. Chung, and A.Y. Ng, “3-D Depth Reconstruction from a Single Still Image,” Int’l J. Computer Vision, vol. 76, pp. 53-69, 2007.

A. Saxena, S. Chung, and A. Ng, “Learning Depth from Single Monocular Images,” Proc. Neural Information Processing Systems, 2005.

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,” Psychological Rev., vol. 113, no. 4, pp. 766-786, 2006.

R. Achanta, S. Hemami, F. Estrada, and S. Susstrunk, “Frequency-Tuned Salient Region Detection,” Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2009.

M. Everingham, A. Zisserman, C.K.I. Williams, and L. Van Gool, “The PASCAL VOC2006 Results,” http://www.pascal-network.org/challenges/VOC/voc2006/results.pdf, 2012.

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/, 2012.

N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection,” Proc. IEEE CS Conf. Computer Vision and Pattern Recognition, 2005.

A. Saxena, J. Driemeyer, J. Kearns, and A.Y. Ng, “Robotic Grasping of Novel Objects,” Proc. Neural Information Processing Systems, 2006.

A. Saxena, J. Driemeyer, and A.Y. Ng, “Robotic Grasping of Novel Objects Using Vision,” Int’l J. Robotics Research, vol. 27, no. 2, pp. 157-173, 2008.

L. Fei-Fei and P. Perona, “A Bayesian Hierarchical Model for Learning Natural Scene Categories,” Proc. IEEE CS Conf. Computer Vision and Pattern Recognition, 2005.

V. Hedau, D. Hoiem, and D. Forsyth, “Recovering the Spatial Layout of Cluttered Rooms,” Proc. 12th IEEE Int’l Conf. Computer Vision, 2009.

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, 2012.

C. Li, T. Wong, N. Xu, and A. Saxena, “FECCM for Scene Understanding: Helping the Robot to Learn Multiple Tasks,” Proc. IEEE Int’l Conf. Robotics and Automation, http://chenlab.ece.cornell.edu/projects/FECCM/, 2011.

Congcong Li received the BE and MS degrees in 2005 and 2007, respectively, from Tsinghua University, China. She started working toward the PhD degree at Carnegie Mellon University, from 2007 and moved to the School of Electrical and Computer Engineering, Cornell University, in 2009. Her research interests include computer vision, machine learning, and image analysis. She is a student member of the IEEE.

Adarsh Kowdle received the BE degree from Rashthreya Vidyalaya College of Engineering, India, in 2007. He started working toward the PhD degree in the School of Electrical and Computer Engineering at Cornell University in 2009, before which he spent a semester at Carnegie Mellon University. From June 2007 to July 2008, he worked as an engineer with Ittiam Systems. His research interests include interactive computer vision algorithms and machine learning. He is a student member of the IEEE.

Ashutosh Saxena received the BTech degree in 2004 from the Indian Institute of Technology (IIT) Kanpur, and the MS and PhD degrees in 2006 and 2009, respectively, from Stanford University. He is an assistant professor in Computer Science Department at Cornell University. His research interests include machine learning, robotics, and computer vision. He has developed Make3D (http://make3d.cs.cornell.edu), an algorithm that converts a single photograph into a 3D model. Tens of thousands of users have 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 a substantial amount of attention in the popular press, including the front page of the New York Times, BBC, ABC, New Scientist Discovery Science, and Wired Magazine. He has won best paper awards at 3DRR and IEEE ACE. He was also a recipient of a National Talent Scholar award in India, and a Sloan research fellowship in 2011. He is a member of the IEEE.

Tsuhan Chen received the BS degree in electrical engineering from the National Taiwan University, Taipei, in 1987 and the MS and PhD degrees in electrical engineering from the California Institute of Technology, Pasadena, in 1990, respectively, from Stanford University. He has been with the School of Electrical and Computer Engineering, Cornell University, Ithaca, New York, since January 2009, where he is a professor and director. From October 1997 to was with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, as a professor and associate department head. From August 1993 to October 1997, he worked at AT&T Bell Laboratories, Holmdel, New Jersey. He coedited a book titled Multimedia Systems, Standards, and Networks (Marcel Dekker, 2000). He served as an editor-in-chief for the IEEE Transactions on Multimedia in 2002-2004. He also served on the editorial board of IEEE Signal Processing Magazine and as an associate editor for the 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 from the California Institute of Technology in 1993. He was a recipient of the US National Science Foundation Career Award (NSF) from 2000 to 2003. He received the Benjamin Richard Teare Teaching Award in 2006, and theEta Kappa Nu Award for Outstanding Faculty Teaching in 2007. He was elected to the Board of Governors, IEEE Signal Processing Society, 2007-2009, and was a distinguished lecturer of the IEEE Signal Processing Society from 2007-2008. He is a member of the Phi Tau Phi Scholastic Honor Society and a fellow of the IEEE.