Towards Holistic Scene Understanding: Feedback Enabled Cascaded Classification Models

Congcong Li, Adarsh Kowdle, Ashutosh Saxena, Tsuhan Chen
Cornell University, Ithaca, NY.
{cl758,apk64}@cornell.edu,
asaxena@cs.cornell.edu, tsuhan@ece.cornell.edu

Abstract

In many machine learning domains (such as scene understanding), several related sub-tasks (such as scene categorization, depth estimation, object detection) operate on the same raw data and provide correlated outputs. Each of these tasks is often notoriously hard, and state-of-the-art classifiers already exist for many sub-tasks. It is desirable to have an algorithm that can capture such correlation without requiring to make any changes to the inner workings of any classifier.

We propose Feedback Enabled Cascaded Classification Models (FE-CCM), that maximizes the joint likelihood of 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 what error modes to focus on. We show that our method significantly improves performance in *all* the sub-tasks in two different domains: (i) scene understanding, where we consider depth estimation, scene categorization, event categorization, object detection, geometric labeling and saliency detection, and (ii) robotic grasping, where we consider grasp point detection and object classification.

1 Introduction

In many machine learning domains, several sub-tasks operate on the same raw data to provide correlated outputs. Each of these sub-tasks are often notoriously hard and state-of-the-art classifiers already exist for many of them. In the domain of scene understanding for example, several independent efforts have resulted in good classifiers for tasks such as scene categorization, depth estimation, object detection, etc. In practice, we see that these sub-tasks are coupled—for example, if we know that the scene is indoors, it would help us estimate depth more accurately from that single image. In another example in the robotic grasping domain, if we know what object it is, then it is easier for a robot to figure out how to pick it up. In this paper, we propose a unified model which 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 machine learning domains.

Recently, several approaches have tried to combine these different classifiers for related tasks in vision [19, 25, 35]; however, most of them tend to be ad-hoc (i.e., a hard-coded rule is used) and often 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. [17] 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 would 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, to estimate the depth well in that region by sacrificing some regions in the bottom may help an image to be categorized to the correct category. 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 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 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 tasks that have different but correlated outputs.

In extensive experiments, we show that our method achieves significant improvements in the performance of *all* the sub-tasks in two different domains: (i) scene understanding, where we consider six tasks: depth estimation, object detection, scene categorization, event categorization, geometric labeling and saliency detection, and (ii) robotic grasping, where we consider two tasks: grasp point detection and object classification.

The rest of the paper is organized as follows. We discuss the related works in Section 2. We describe our FE-CCM method in Section 3 followed by the implementation of the classifiers in Section 4. We present the experiments and results in Section 5. We finally conclude in Section 6.

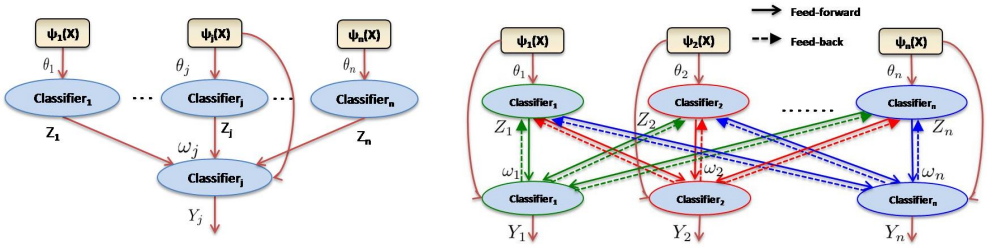
2 Related Work

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 [16, 12, 6]. 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.

There has been a huge body of work in the area of sensor fusion where classifiers 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 [21]. 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.

The idea of improving classification performance by combining outputs of many classifiers is used in methods such as Boosting [13], where many weak learners are combined to obtain a more accurate classifier; this has been applied tasks such as face detection [4, 40]. However, unlike the CCM framework which focuses on contextual benefits, their motivation was computational efficiency. Tu [39] used pixel-level label maps to learn a contextual model for pixel-level labeling, through a cascaded classifier approach, but such works considered only the interactions between labels of the same type.

While the above combine classifiers to predict the *same* labels, there are a group of works that combine classifiers, and use them as components in large systems. Kumar and Hebert [23] 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 [36] 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



(a) (b)

Figure 1: Combining related classifiers using the proposed FE-CCM model ($\forall i \in \{1, 2, \ldots, n\}$ $\Psi_i(X)$ = 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): (a) Cascaded classification model (CCM) where the output from the previous stage of the classifier is used in the subsequent stage along with image features. The model optimizes the output of each $Classifier_j$ on the second stage independently; (b) Proposed Feed-back enabled cascaded classification model (FE-CCM), where 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)

workings of the individual classifiers. Structured learning (e.g., [38]) could also be a viable option for our setting, however, they need a fully-labeled dataset which is not available in vision 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., [29, 37, 2]). Saxena et. al. manually designed the terms in an MRF to combine depth estimation with object detection [34] and stereo cues [33]. Sudderth et al. [35] used object recognition to help 3D structure estimation. Hoiem et. al. [19] 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. [25, 24] 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 tasks. This limits the generality of the approaches in introducing new tasks easily or being applied to other domains.

There is also a large body of work in the areas of deep learning, and we refer the reader to Bengio and LeCun [3] for a nice overview of deep learning architectures and Caruana [5] for multitask learning with shared representation. While most works in deep learning (e.g., [15, 18, 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. [18] used unsupervised learning to obtain an initial configuration of the parameters. This provides a good initialization and hence their multi-layered architecture does not suffer from local minimas during optimization. At a high-level, we can also look at our work as a multi-layered architecture (where each node typically produces complex outputs, e.g., labels over the pixels in the image); and initialization in our case comes from existing state-of-the-art individual classifiers. Given this initialization, our training procedure finds parameters that (consistently) improve performance across all the sub-tasks.

3 Feedback Enabled Cascaded Classification Models

We will describe the proposed model for combining and training the classifiers in this section.

We consider related subtasks denoted by Classifier_i, where $i \in \{1, 2, \ldots, n\}$ for a total of n tasks (Figure 1). Let $\Psi_i(X)$ correspond to the features extracted from image X for the Classifier_i. Our cascade is composed of two layers, where the outputs from classifiers on the first layer go as input into the classifiers in the second layer. We do this by appending all the outputs from the first layer to the features for that task. θ_i represents the parameters for the first level of Classifier_i with output Z_i, and ω_i represents the parameters of the second level of Classifier_i with output Y_i.

We model the conditional joint log likelihood of all the classifiers, i.e., $\log P(Y_1, Y_2, \ldots, Y_n|X)$, where X is an image belonging to training set Γ.

$$\log \prod_{X \in \Gamma} P(Y_1, Y_2, \ldots, Y_n|X; \theta_1, \theta_2, \ldots, \theta_n, \omega_1, \omega_2, \ldots, \omega_n) \tag{1}$$

During training, $Y_1, Y_2, \ldots, Y_n$ are all observed (because the ground-truth labels are available). However, $Z_1, Z_2, \ldots, Z_n$ (output of layer 1 and input to layer 2) are hidden, and this makes training of each classifier as a black-box hard. Heitz et al. [17] 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 $Z_1, Z_2, \ldots, Z_n$ for training the classifiers.

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, which would result in the second layer's output $(Y_1, Y_2, \ldots, Y_n)$ to become the best. Therefore, we expand Equation 1 as follows, using the independencies represented by the directed graphical model in Figure 1(b).

$$= \sum_{X \in \Gamma} \log \sum_{Z_1,\ldots,Z_n} P(Y_1, \ldots, Y_n, Z_1, \ldots, Z_n | X; \theta_1, \ldots, \theta_n, \omega_1, \ldots, \omega_n) \tag{2}$$

$$= \sum_{X \in \Gamma} \log \sum_{Z_1,\ldots,Z_n} \prod_{i=1}^{n} P(Y_i | \Psi_i(X), Z_1, \ldots, Z_n; \omega_i) P(Z_i | \Psi_i(X); \theta_i) \tag{3}$$

However, the summation inside the log makes it difficult to learn the parameters. Motivated by the Expectation Maximization [8] 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. 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 [17] which in our formulation is equivalent to *fixing* the latent variables to ground-truth permanently (thus highlighting the impact of the feedback).

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 [17], 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

$$\underset{\theta_1,\ldots,\theta_n,\omega_1,\ldots,\omega_n}{\text{maximize}} \sum_{X \in \Gamma} \log \prod_{i=1}^{n} P(Y_i | \Psi_i(X), Z_1, \ldots, Z_n; \omega_i) P(Z_i | \Psi_i(X); \theta_i) \tag{4}$$

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:

$$\underset{\omega_i}{\text{maximize}} \sum_{X \in \Gamma} \log P(Y_i | \Psi_i(X), Z_1, \ldots, Z_n; \omega_i) \tag{5}$$

$$\underset{\theta_i}{\text{maximize}} \sum_{X \in \Gamma} \log P(Z_i | \Psi_i(X); \theta_i) \tag{6}$$

Note that the optimization problem nicely breaks down into the sub-problems of training the individual classifier for the respective sub-tasks. Depending on the specific form of the classifier used for the sub-task (see Section 4 for our implementation), we can use the appropriate training method for each of them. For example, we can use the same training algorithm as the original black-box classifier. Therefore, we consider the original classifiers as black-box and we do not need any low level information about the particular tasks or knowledge of the inner workings of the classifier.

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, [26]). Using Equation 3, we get the following optimization problem for the feed-back step:

$$\begin{aligned} &\underset{Z_1,\ldots,Z_n}{\text{maximize}} \log P(Y_1, \ldots, Y_n, Z_1, \ldots, Z_n | X; \theta_1, \ldots, \theta_n, \omega_1, \ldots, \omega_n) \\ \Leftrightarrow &\underset{Z_1,\ldots,Z_n}{\text{maximize}} \sum_{i=1}^{n} \log P(Z_i | \Psi_i(X); \theta_i) + \log P(Y_i | \Psi_i(X), Z_1, \ldots, Z_n; \omega_i) \end{aligned} \tag{7}$$

This maximization problem requires that we have access to the characterization of the individual black-box classifiers in a probabilistic form. While at the first blush this may seem asking a lot, our method can even handle classifiers for which 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 [14]). Parameters of the Gaussians are empirically estimated when the actual model is not available.

In some cases, the classifier log-likelihoods in the problem in Equation 7 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 by using a gradient descent (or any similar method).

Figure 2: Results showing improvement using the proposed model. 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).

Inference. Our FE-CCM is a directed model and inference in these models is straight-forward. Maximizing the conditional log likelihood $P(Y_1, Y_2, \ldots, Y_n|X)$ corresponds to performing inference over the first layer (using the same inference techniques for the respective black-box classifiers), followed by inference on the second layer.

Sparsity and Scaling with a large number of tasks. In Equations 4 we use weight decay (with L-1 penalty on the weights, $||\omega||_1$) to enforce sparsity in the ω's. With a large number of sub-tasks, the number of the weights in the second layer increases, and our sparsity term results in a few non-zero connections between sub-tasks that are active.

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 showed our formulation for the general case, where we use Γ_i as the dataset that has labels for i^{th} task. Now, we maximize the joint likelihood over all the datapoints, i.e., $\log \prod_{i=1}^{n} \prod_{X \in \Gamma_i} P(Y_1, \ldots, Y_n|X)$. Equation 3 reduces to maximizing the terms below, which is solved using equations in Section 3 with corresponding modification

$$\sum_{i=1}^{n} \lambda_i \sum_{X \in \Gamma_i} \log \sum_{Z_1, \ldots, Z_n} P(Y_i|\Psi_i(X), Z_1, \ldots, Z_n; \omega_i) \prod_{j=1}^{n} P(Z_j|\Psi_j(X); \theta_j) \tag{8}$$

Here λ_i is the tuning parameter that balances the amount of data in different datasets ($n = 6$ in our experiments).

4 Scene Understanding: Implementation

Here we briefly describe the implementation details for our instantiation of FE-CCMs for scene understanding.[1] 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 comparison will still be against the state-of-the-art models for the respective sub-tasks (and on the same standard respective datasets) in Section 5.

In our preliminary work [22], where we optimized for each target task independently, we considered four vision tasks: scene categorization, depth estimation, event categorization and saliency detection. Please refer to Section 4 in [22] for implementation details. In this work, we add object detection and geometric labeling, and jointly optimize all six tasks.

Scene Categorization. For scene categorization, we classify an image into one of the 8 categories defined by Torralba et. al. [28]: tall building, inside city, street, highway, coast, open-country, mountain and forest. We define the output of a scene classifier to be a 8-dimensional vector with each element representing the score for each category. We evaluate the performance by measuring the accuracy of assigning the correct scene label to an image on the MIT outdoor scene dataset [28].

Depth Estimation. For the single image depth estimation task, we want to estimate the depth $d \in \mathcal{R}_+$ of every pixel in an image (Figure 2a). 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 [30, 31].

[1] Space constraints do not allow us to describe each sub-task in detail here, but please refer to the respective state-of-the-art algorithm. Note that the power of our method is in *not* needing to know the details of the internals of each sub-task.

Event Categorization. For event categorization, we classify an image into one of the 8 sports events as defined by Li et. al. [24]: bocce, badminton, polo, rowing, snowboarding, croquet, sailing and rock-climbing. We define the output of a event classifier to be a 8-dimensional vector with each element representing the log-odds score for each category. For evaluation, we compute the accuracy assigning the correct event label to an image.

Saliency Detection. Here, we want to classify each pixel in the image to be either salient or non-salient (Figure 2c). We define the output of the classifier as a scalar indicating the saliency confidence score of each pixel. We threshold this saliency score to determine whether the point is salient $(+1)$ or not (-1). For evaluation, we compute the accuracy of assigning a pixel as a salient point.

Object Detection. We consider the following object categories: car, person, horse and cow. A sample image with the object detections is shown in Figure 2b. We use the train-set and test-set of PASCAL 2006 [9] for our experiments. Our object detection module builds on the part-based detector of Felzenszwalb et. al. [10]. We first generate 5 to 100 candidate windows for each image by applying the part-based detector with a low threshold (over-detection). We then extract HOG features [7] on every candidate window and learn a RBF-kernel SVM model as the first layer classifier. The classifier assigns each window a $+1$ or -1 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.

Geometric labeling. The geometric labeling task refers to assigning each pixel to one of three geometric classes: support, vertical and sky (Figure 2d), as defined by Hoiem et. al. [20]. We use the dataset and the algorithm by [20] 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 about 100 segments/image. For the second-layer, we learn a logistic model over the a feature vector which is constituted by the outputs of all first-level tasks and the features used in the first layer. For evaluation, we compute the accuracy of assigning the correct geometric label to a pixel.

5 Experiments and Results

The proposed FE-CCM model is a unified model which jointly optimizes for all 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 two different domains: scene understanding and robotic grasping. We use the *same* proposed algorithm in both domains. For each of the sub-task 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 4) 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. [17], 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 (target specific): Here, we train a specific FE-CCM for each sub-task, by using cross-validation to estimate λ_i's in Equation 8. 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.

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 (% Average precision)				
						Car	Person	Horse	Cow	Mean
Images in testset	1579	400	2688	1000	300	2686				
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$0.6)	62.3	36.8	38.8	40.0	44.5
State-of-the-art model (reported)	73.4 Li [24]	16.7 (MRF) Saxena [31]	83.8 Torralba [27]	82.5 ($\pm$0.2) Achanta [1]	88.1 Hoiem [20]	61.5 Felzenswalb et. al. [11] (base)	36.3	39.2	40.7	44.4
CCM [17] (our implementation)	73.3 ($\pm$1.6)	16.4 ($\pm$0.4)	83.8 ($\pm$0.6)	85.6 ($\pm$0.2)	87.0 ($\pm$0.6)	62.2	37.0	38.8	40.1	44.5
FE-CCM (unified)	**74.3 ($\pm$0.6)**	**15.5 ($\pm$0.2)**	**85.9 ($\pm$0.3)**	**86.2 ($\pm$0.2)**	**88.6 ($\pm$0.2)**	**63.2**	**37.6**	**40.1**	40.5	**45.4**
FE-CCM (target specific)	**74.7 ($\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**

5.1 Scene Understanding

Datasets: The datasets used are mentioned in Section 4, and the number of test images in each dataset is 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 [9] dataset (1277 train and 2686 test images respectively).

Results and discussion:

To quantitatively evaluate our method for each of the sub-tasks, we consider the metrics appropriate to each of the six tasks in Section 4. Table 1 shows that FE-CCM not only beats state of 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 [24, 31] or by using adhoc methods to implicitly use information from other tasks [20]. 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 observe that our target-specific FE-CCM, which is optimized for each task independently achieves the best performance, and this is a more fair comparison to the state-of-the-art because each state-of-the-art model is trained specifically to the respective task. Furthermore, Table 1 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 Figure 2. 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. More visual results are provided in the supplementary material.

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 3-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 3-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 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.

5.2 Robotic Grasping

In order to show the applicability of our FE-CCM to problems across different machine learning experiments, we also considered the problem of a robot autonomously grasping objects. Given an image and a depthmap, the goal of the learning algorithm is to select a point at which to grasp the

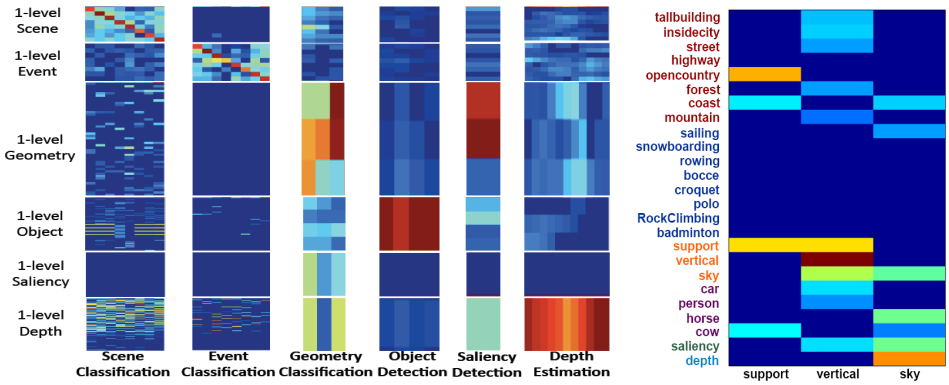


Figure 3: (Left) The *absolute* values of the weight vectors for second-level classifiers, i.e. ω. Each column shows the contribution of the various tasks towards a certain task. (Right) Detailed illustration of the *positive* values in the weight vector for a second-level geometric classifier. (Note: Blue is low and Red is high)

object (this location is called grasp point, [32]). It turns out that different categories of objects could have different strategies for grasping, and therefore 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. [32] which spans 6 object categories and also includes an aligned pixel level depth map for each image. For grasp point detection, we use a regression over features computed from the image [32]. The output of the regression is a score for each point giving the confidence of the point being a good 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.

Results: We evaluate our algorithm on dataset published in [32], and perform cross-validation to evaluate the performance on each task. 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 4 show our robot grasping an object using our algorithm.

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**

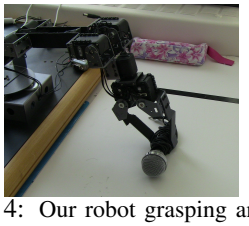


Figure 4: Our robot grasping an object using our algorithm.

6 Conclusions

We propose a method for combining existing classifiers for different but related tasks. 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.

We consider two domains: scene understanding and robotic grasping. Our unified model (a single FE-CCM trained for all the sub-tasks in that domain) 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.

Acknowledgements

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

References

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

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

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

[4] 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*, 77(1-3):65–86, 2008.

[5] R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[6] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, 2008.

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

[8] 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*, 39(1):1–38, 1977.

[9] M. Everingham, A. Zisserman, C. K. I. Williams, and L. Van Gool. The pascal voc2006 results.

[10] 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/.

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

[12] Y. Freund and R. E. Schapire. Cascaded neural networks based image classifier. In *ICASSP*, 1993.

[13] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT*, 1995.

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

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

[16] L. Hansen and P. Salamon. Neural network ensembles. *PAMI*, 12(10):993–1001, 1990.

[17] G. Heitz, S. Gould, A. Saxena, and D. Koller. Cascaded classification models: Combining models for holistic scene understanding. In *NIPS*, 2008.

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

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

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

[21] J. Kittler, M. Hatef, R. P. Duin, and J. Matas. On combining classifiers. *PAMI*, 20:226–239, 1998.

[22] 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.

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

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

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

[26] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. *Learning in graphical models*, 89:355–368, 1998.

[27] A. Oliva and A. Torralba. Mit outdoor scene dataset. http://people.csail.mit.edu/torralba/code/spatialenvelope/.

[28] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42:145–175, 2001.

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

[30] A. Saxena, S. H. Chung, and A. Y. Ng. Learning depth from single monocular images. In *NIPS*, 2005.

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

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

[33] A. Saxena, J. Schulte, and A. Y. Ng. Depth estimation using monocular and stereo cues. In *IJCAI*, 2007.

[34] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE PAMI*, 30(5), 2009.

[35] 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.

[36] C. Sutton and A. McCallum. Joint parsing and semantic role labeling. In *CoNLL*, 2005.

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

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

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

[40] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004.

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