Model Recommendation: Generating Object Detectors from Few Samples

Yu-Xiong Wang and Martial Hebert
Robotics Institute, Carnegie Mellon University
{yuxiongw, hebert}@cs.cmu.edu

Abstract

In this paper, we explore an approach to generating detectors that is radically different from the conventional way of learning a detector from a large corpus of annotated positive and negative data samples. Instead, we assume that we have evaluated "off-line" a large library of detectors against a large set of detection tasks. Given a new target task, we evaluate a subset of the models on few samples from the new task and we use the matrix of models-tasks ratings to predict the performance of all the models in the library on the new task, enabling us to select a good set of detectors for the new task. This approach has three key advantages of great interest in practice: 1) generating a large collection of expressive models in an unsupervised manner is possible; 2) a far smaller set of annotated samples is needed compared to that required for training from scratch; and 3) recommending models is a very fast operation compared to the notoriously expensive training procedures of modern detectors. (1) will make the models informative across different categories; (2) will dramatically reduce the need for manually annotating vast datasets for training detectors; and (3) will enable rapid generation of new detectors.

1. Motivation

> Essentially, all models are wrong, but some are useful.
>
> — George E. P. Box; Norman R. Draper, 1987 [6]

Over the past few decades, there has been much progress in designing effective object detectors, especially when enough data are available [14, 33, 36, 56, 37, 15, 48]. For example, flagship techniques such as Deformable Part Models (DPMs) [14] via sliding-window fashion are best suitable for semi-rigid objects, while Exemplar-SVMs [33] with simple linear SVMs are highly category and instance specific; the recent top performing detection systems instead favor bottom-up region proposals [52, 10, 15] which tend to perform well for non-rigid objects. Along with these, more powerful appearance models, feature learning mechanisms beyond standard hand-engineered features such as R-CNN [15] and GoogLeNet [48] are emerging.

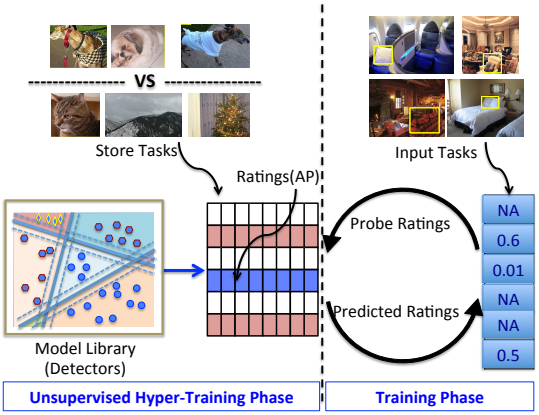

Figure 1. During unsupervised hyper-training phase, a large library of object detectors informative across categories is generated. Their ratings on different detection tasks are recorded to form a ratings store. For a new target task or category and using ratings of a small probe set of detectors on its input task with limited samples, recommendations are made by collaborative filtering. A usable object detector for this new task is thus rapidly generated as single or ensemble of the recommended models.

All of these modern detectors have in common the same supervised training framework in which a large annotated dataset is required and in which training from scratch is restarted for a new task, *e.g.*, a new category. In practice, however, it might be difficult to produce enough data for a new task. Moreover, in many applications it is desirable to *rapidly* train a new detector for a new task, something that is typically not possible in any of these current approaches which require expensive training iterations. Thus, our objective is to quickly generate good models for a detection task given a small number of labeled samples.

Our approach is based on the informal intuition that, given a very large set of models, it is likely that some of the models would have good performance on a new detection task, as stated in the *infinite monkey theorem* [1]. More formally, inspired by this intuition, we explore a new approach, which is based on the observation that, while it may

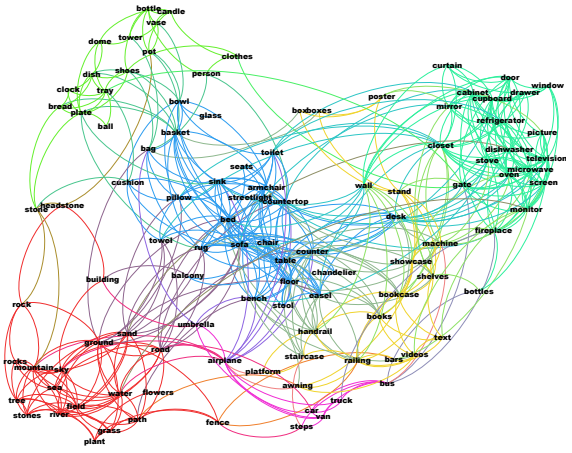

Figure 2. Continuous Category Space Discovery. We measure the similarity between the 107 categories on SUN by their number of shared PBC models within the top-200 ranked ones, and visualize them using ForceAtlas [5, 17]. The edge between two category nodes indicates that they have at least 10 models in common. Although the models are generated entirely unsupervisedly on PASCAL, it is interesting to note that visually or functionally similar categories are naturally grouped together: such as the green cluster of monitor, stove, microwave, oven, dishwasher; red cluster of sea, river, field, sky; and purple cluster of car, bus, van, truck, *etc*.

be hard (impossible) to generate enough training data for a new task, it may be easy to generate a large library of models and evaluate them *off-line* on a large set of tasks. Given this data on the performance of many models on many tasks, it may be possible to guess which of these stored models are best suited for a novel target task by evaluating them on that new task. The hope of course is that we can do so with a smaller set of samples of the target task than would be necessary for training from scratch. In addition to (hopefully) requiring a much smaller sample set, this approach could be far more efficient because it merely *recommends* models, which have been generated off-line.

A natural question then is why should such an approach be possible? The first observation is that, while any *specific* detector cannot generalize well across tasks, in a large-scale library, however, it is likely that one of the library models happens to be tuned with the similar conditions as the new target task. Combining multiple such models into a single new model may perform well on the new task. This would be true especially when considering the shared properties across instances and categories [30, 39, 12, 20]. This is not sufficient, however, as we are still faced with the problem of selecting the right models out of the library. A naïve approach would be to evaluate each model from the library on the input task and select the one(s) that perform the best. Unfortunately, this direct approach typically performs poorly because of the limited data available in the input task. More precisely, the evaluation scores of the library model on the input task might be noisy enough that the ranking of the models is not reliable.

The second observation then is that, if the library and the set of tasks are large enough, there might be enough correlation between the models that it is possible to predict the performance of the models on the target tasks by using the *entire* combined experience with the model library and the tasks. This is similar to the approach taken in recommender systems [25] in which a matrix of ratings of items (the analog of our models) by users (the analog of our tasks) is used to predict the ratings that a new user (the target task) would generate, *i.e*., predict the top ranked items (the analog of the best models for the target task). Inspired by prior work in the context of recommending action classifiers [35, 34], our goal is to explore how this approach can be used in *detection* tasks—a more general and challenging recognition scenario. While we share the same general idea of using recommender system for vision tasks, the specific technical approaches here are quite different from those in [35, 34].

The general setup is thus as follows: We first build a large library of object detectors and we record their performance or "ratings" on a large set of detection tasks, which we call the "ratings store". In order to distinguish this phase from the traditional training phase, and by analogy with training hyper-parameters, we call the process of generating and evaluating large library of models *off-line* as "hyper-training". Given a new target task, recommendations are made by trying, or rating, a small subset, called the probe set, of detectors on the input task with few samples. That small set of ratings, along with the ratings of all the detectors in the ratings store, is used to predict the ratings of all the models on the target task. We then select models based on the recommendations and use them for the new task. This thus becomes our new formulation of object detection, which we term as *model recommendation for object detection* as shown in Fig. 1. Now, the model recommendation system provides a succinct way to combine both large-scale models and big visual data into a joint task-model space. By directly manipulating in the new space, the experience of acquiring models becomes a procedure similar to shopping for finished products in a store.

Our contribution is thus four-fold: First, we show how such a recommendation setup can be operationalized using collaborative filtering techniques (Sections 2 and 4), including an analysis of its operation on a large-scale, controlled experiment (Section 5). Second, we detail how tasks are defined and detectors are generated in the off-line stage. Third, we show how a universal detector library, based on combining *unsupervised* predictable discriminative binary codes (PBCs) classifiers and convolutional neural networks (CNNs) features, is generated without a bias to a particular set of categories, and we show how it is particularly effective for populating the ratings store in this setup (Sections 6). New detectors are thus obtained rapidly on novel

classes with few samples *without* conventional supervised training involved. Finally, another interesting finding of this large-scale model system is a continuous category space discovered by model sharing as shown in Fig. 2 and explained in Section 6.2.2.

2. Terminology and Approach Overview

A *task* $T_j = \{(x_{j1}, y_{j1}), (x_{j2}, y_{j2}), ...\}$ is assumed to be a self-contained object detection problem, where $\{x_{j1}, x_{j2}, ...\}$ are input images, and $\{y_{j1}, y_{j2}, ...\}$ are the corresponding annotations indicating labels and bounding boxes for objects of interest. Note that annotations might have different interpretations across tasks. Moreover, the image samples associated with a task need not be unique; the same data sample might be shared across many tasks (*e.g.*, if one dataset is used for different tasks).

Models we are interested in here are object detectors. They can be pre-trained detectors with no free parameters, or they can be detectors trained on different data sources. They can be quite strong models highly tuned for certain tasks, or they can be generic weak detectors.

The *rating* of a model M_i on a task T_j is a number describing how well the model performs on that task, which is denoted by $R(M_i; T_j) = r_{ij}$, and is restricted to the range $[0, 1]$. For example, for the detection task, we follow the performance evaluation procedure of the PASCAL VOC Challenge [13]. A detection is considered correct when the area of overlap (measured by intersection-over-union, IoU) between the predicted bounding box and the ground-truth bounding box exceeds 50%, and the precision/recall curve is computed accordingly. Since a single number is needed as rating, the average precision (AP) is chosen to summarize the shape of the precision/recall curve.

The *ratings store* is a matrix R, of the ratings of the models in the library on different tasks, where rows correspond to models, and columns correspond to tasks. The ratings store records the performance of models on tasks generated off-line during the hyper-training phase, and encodes the performance distribution of models on different tasks. The matrix is $n \times m$, where n is the number of models in the library, and m is the number of store tasks. We assume that R is complete, *i.e.*, every model has been rated on every task in the store.

Finally, the task in which we are actually interested is a *target (or hidden) task*. The data in the *input task* is limited to a few samples from this target task. For instance, the target task is to detect a certain type of cat; however, for its input task, we only have access to a limited number of training samples, *e.g.*, on the order of ten samples. We hope to recommend models that work well on the target task. The available ratings are the ones from the probe set on the input task, which are both limited and noisy. When this set of ratings on the input task is fed into a model recommendation system, the returned predictions are a full and de-noised version for the target task. Recommender systems relevant to our scenario are based on well-established collaborative filtering techniques [25].

Due to the complexity of real-world scenarios, recommending a single best model and hoping it will work well on the target task might not be realistic. One extension is to jointly recommend sets of models as in ensemble learning [58] and mixture of experts [19]. The simplest approach is to use the top-k recommendations, which simply selects the top-k models based on their predicted ratings from model recommendation. However, this obvious strategy can potentially recommend a highly redundant set. One alternative is to recommend a diverse set of top performing models [45]. Once a set of models is selected, they are combined by using a task-dependent fusion strategy, *e.g.*, by training weights for the models on the input task.

To sum up, we start with a large collection of models and images, from which we build a ratings store during hyper-training. Then, during training given a new input task and a probe set on that task, we use collaborative filtering techniques to predict the ratings of all the models on the hidden task and to return a single or ensemble of models with the highest predicted ratings as the recommended models.

3. Related Work

Intuitively, in this approach we generate a new model by learning from experience, *i.e.*, from the matrix of evaluations of models on tasks, instead of learning from supervised data as is normally the case. The generated model library can be also viewed as a prior or regularization with respect to the common visual knowledge. In previous work, examples of transfer of prior experience to a new task include concept drift [51], domain adaption, transfer learning [40] (*e.g.*, sparse prototype representations [43], hypothesis transfer learning [27, 49], regularized SVM [4]), multi-task learning (*e.g.*, rank-reduced shared classification model [2, 41]), concept modeling in the field of multimedia (*e.g.*, LSCOM [38]), intermediate representation based on learning classifiers on related tasks (*e.g.*, Object Bank [29, 57], Classemes [50]), which address a different scenario and often require extensive supervised retraining on the new target task.

In terms of transfer scenarios, transductive transfer learning, *e.g.*, domain adaptation, assumes different domains yet single task [40], while we consider totally different tasks. In contrast to inductive transfer learning, *e.g.*, multi-task learning, which learns all of the source and target tasks simultaneously [2], the construction of our model library and ratings store is completed without having access to the target task in advance. As for transfer techniques, most of the existing approaches transfer either on a data or instance level, *e.g.*, importance sampling and re-weighting, or on a

feature or parameter level, *e.g.*, discovering shared feature representation, parameters, or priors across related tasks [3], which often require data reusing in the source domains and extensive supervised retraining on the target task. Instead, transfer happens directly on a model level in our system; the desirable information is maintained in the model library and ratings store while the original data are discarded.

The proposed system thus provides a general principle and basic framework in which several crucial issues for transfer including 1) what to transfer, 2) how to transfer, and 3) when to transfer [40] could be tackled together. For instance, the source models in transfer learning are usually well-trained categorical classifiers [49, 4]. However, none of the models in our library is required to be specifically designed for transfer learning. They can be learned separately and different types of detectors are also possible. In particular, we generate a universal detector library without any supervision, which frees the model library from ties to a specific set of categories. If we view each base model as a source domain, their scale is enormous and the issue of selecting the right domain to transfer (including the scalability) also arises which was normally not addressed before.

4. Collaborative Filtering

Based on the probe set ratings and the ratings store, collaborative filtering techniques predict the ratings of the entire library. We use collaborative filtering techniques based on matrix factorization, which assume a low-rank approximation to the ratings store that naturally embeds both tasks and models to a joint latent factor space of dimensionality d, such that task-model interactions are modeled as inner products in that space [25, 24]. Although the rating distribution of model recommendation might be different from that of the typical consumer recommender system, the ratings store still has exchangeability properties—arrays of random variables whose distributions are invariant to permutations of rows and of columns, which makes it statistically justified to use a factor model that implicitly encodes the Aldous-Hoover theorem for exchangeable matrices [23, 22].

In this approach, we associate each model M_i with a vector $u_i \in \mathbb{R}^d$, and each task T_j with a vector $v_j \in \mathbb{R}^d$. For a given model M_i, the elements of u_i measure the extent to which the model possesses the factors. For a given task T_j, the elements of v_j measure how well models with the corresponding factors will perform on the task. The interaction between M_i and T_j, *i.e.*, the overall performance of that model on the task, is then characterized by the dot product of u_i and v_j. The estimate of rating r_{ij} of model M_i on task T_j is approximated as

$$\widehat{r}_{ij} = u_i^T v_j. \tag{1}$$

After the recommender system infers u_i and v_j from rating patterns, it can easily predict the rating a model will give to any task by using Eq. (1). The crucial issue is how to transform each model and task into vectors u_i, v_j.

Many matrix factorization techniques can be considered in this context, such as restricted Boltzmann machines [47], sparse coding [32, 31], and maximum margin matrix factorization optimizing directly for ranking scores [54]. Here we limit ourselves to direct factorization approaches which are sufficient for our purpose. Specifically, we consider approaches based on singular value decomposition (SVD) and non-negative matrix factorization (NMF), which we describe briefly in this section.

4.1. Factorization Techniques based on SVD

One simple way to identify the latent semantic model and task factors is by singular value decomposition (SVD) to decompose the rating matrix R into two factor matrices. In practice, the distribution of ratings may be significantly biased: some models may produce systematically higher ratings than others, and some tasks may be systematically easier than others. Hence, it is necessary to estimate the portion of rating values that individual model or task biases can explain [25]. A first-order approximation of the rating r_{ij} is introduced to identify the biases as $b_{ij} = \mu + q_i + p_j$, where μ denotes the global average rating, and the parameters q_i and p_j indicate the observed deviations of model M_i and task T_j, respectively, from the average. The biases can be easily estimated from the ratings store as in [25, 35] and used to modify Eq. (1) as

$$\widehat{r}_{ij} = \mu + q_i + p_j + u_i^T v_j. \tag{2}$$

Now, the residual rating, defined as $\overline{r}_{ij} = r_{ij} - \mu - q_i - p_j$, does not remain positive. In this case, we can simply use conventional SVD to obtain the factor matrices. Formally, the residual rating matrix is decomposed as

$$\overline{R} = ESF \approx (E_d S_d) F_d = UV, \tag{3}$$

where d indicates the number of factors, S_d is the $d \times d$ upper left sub-matrix of S, and E_d, F_d are the first d columns of the left-singular vector matrix E, the first d rows of the right-singular vector matrix F, respectively. Hence, the model factor is $U = E_d S_d \in \mathbb{R}^{n \times d}$ with u_i^T as its ith row vector, and the task factor is $V = F_d \in \mathbb{R}^{d \times m}$ with v_j as its jth column vector.

4.2. Factorization Techniques based on NMF

Given that the ratings are all non-negative, an alternative approach is to use non-negative matrix factorization (NMF) [53], which incorporates the non-negativity constraint into the factored matrices. Formally, given the rating matrix $R \in \mathbb{R}_{\geq 0}^{n \times m}$ with non-negative elements, NMF seeks to decompose R into a non-negative $n \times d$ basis matrix U (model factor) and a non-negative $d \times m$ coefficient matrix V (task factor) , such that

$$R \approx UV = \sum_{l=1}^{d} U_{\cdot l} V_{l\cdot}, \tag{4}$$

where $U_{\cdot l}$ is the lth column vector of U while $V_{l\cdot}$ is the lth row vector of V. To learn the two factor matrices, we use the prototypical multiplicative update rules [28].

Now, given the ratings of k probe models for an input task, denoted as r_k, the factor vector of the input task can be estimated by casting as a non-negative least-squares problem with respect to v:

$$\left(\widetilde{U}_k\right) v = r_k, \tag{5}$$

where v is a $d \times 1$ vector, and $\widetilde{U}_k$ is a $k \times d$ sub-matrix of U, whose rows are ratings of probe models on store tasks. v can be solved by fixing $\widetilde{U}_k$ while updating v using the multiplicative update rules. After the factor v of the input task is learned, its ratings for all the models is predicted as

$$\widehat{r} = Uv. \tag{6}$$

5. Recommender System Analysis

Naturally, the first and most important question to answer is whether collaborative filtering could successfully recommend *correct* models. A second question is to elucidate the role of the different design choices involved in this approach. To answer these questions, we designed a controlled experiment with real detection tasks, large-scale data (namely, PASCAL VOC 2007) and full-scale ratings store so that for any of the target tasks one or several of the n models in the library is the correct model to use or at least a reasonable enough approximation. Thus, this is an example of controlled recommender systems whose expected performance is known in advance. This is inspired by the M-closed framework [21] for statistical evaluation.

Of course, in order to support this controlled analysis, this setup is somewhat contrived. In particular, it uses a library of supervised models biased to a particular set of categories. We will describe the actual set of models that we propose to use in a real system in the next section, introducing a new way of generating PBC classifiers in an *unsupervised* manner.

Specifically, we use all the 12,608 ESVMs [33] pretrained for 20 categories on the PASCAL VOC 2007 *trainval* dataset [13] as the model library. We treat these models independently here when using them to make detections. When our target task is defined as detection on the PASCAL VOC 2007 *test* dataset for the same 20 categories, there exists a subset of ESVMs from the corresponding category in the library that work well for the task, which are supposed to be identified by the recommender system.

Note that we use ESVMs as a convenient source for generating a large number of models to test our approach. We do not advocate that using ESVMs is in general the best tool for detection and, in fact, many other types of models could be used in this framework. For example, model recommendation can be naturally applied to other banks of detectors such as ELDAs [16] and Object Bank [29].

5.1. Task Generation

For the store tasks, generally speaking, we could design detection tasks based on specific categories, and group different object instances in the dataset accordingly to generate tasks, but this would bias the system strongly toward these specific categories and prevent it from generalizing to new input tasks. Hence, to demonstrate the performance of the proposed framework, we use purely random tasks instead. Specifically, each task is constructed by randomly selecting 10 images from the PASCAL trainval dataset. We generate 10,000 different tasks in total. The final ratings store is of size $12{,}608 \times 10{,}000$. Since the images within a certain task are random and are therefore not tied to a specific category, a detection for a given task is counted as a positive detection if it intersects the bounding box of the ground-truth annotation of any category. This is distinct from the typical detection task setting, where one always focuses on sets of visually similar objects from the same category.

For the input tasks, we use the PASCAL VOC 2007 test dataset. We view the images containing objects from these 20 categories as 20 hidden tasks, respectively. Then, for each hidden task, we randomly select 10 images to create the corresponding input task. We randomly generate 50 input tasks per hidden task, and report the average performance.

5.2. Experimental Evaluation

There are several design choices in this approach, such as different input tasks, collaborative filtering techniques used, number of factors, size of probe sets, size of recommended models, *etc*. We evaluated the impact of all of these design choices on this ESVM-based setup in which we have predictable performance of the models. To this end, we only consider recommending the *single best* model. The model is selected given the input task, and then evaluated on the hidden task. The results are reported as relative mAP to optimum (the best achievable mAP for single ESVM on the hidden task, which is an upper bound on the performance).

Baselines We compare against two natural baselines: 1) Random Search—randomly pick a model from the probe set; 2) Direct Search—pick the best model from the probe set based on their initial ratings on the input task.

Size of Probe Set We randomly select a subset of models as the probe set. The average performance over 20 categories using SVD is shown in Fig. 3 as function of the size of the probe set. The factor numbers are 32 and 64. Model recommendation works consistently better than direct search as the probe set increases, even when using all

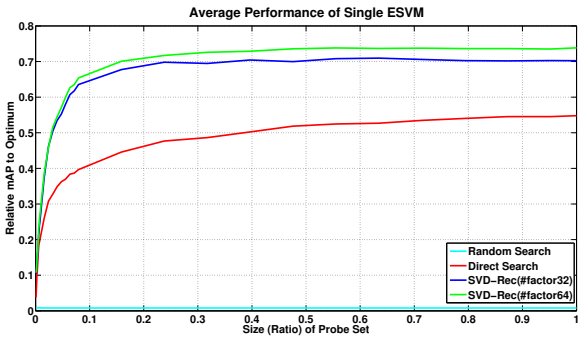

Figure 3. Effect of probe set size for *individual* ESVM across 20 categories on the PASCAL VOC 2007 detection test dataset. X-axis: probe set size (ratio to the size of the entire library). Y-axis: average performance of the recommended model using SVD with different factor numbers (green and blue curves) vs. random search (cyan-blue curve) and direct search (red curve) baselines, reported as relative mAP to optimum, which is the best achievable mAP for single ESVM (upper bound).

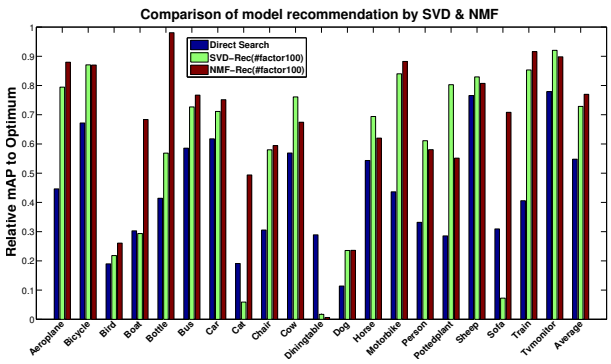

Figure 4. Comparison of different collaborative filtering techniques. X-axis: 20 categories on the PASCAL VOC 2007 dataset. Y-axis: recommendation performance using SVD (green bars) and NMF (red bars) both with factor number 100 vs. direct search baseline (blue bars), reported as relative mAP to optimum. NMF works better than SVD in the majority of cases.

the models as the probe set. Note that the average model performance indicated by random search is quite poor.

Different Collaborative Filtering Techniques The relative performance of SVD and NMF when all the models are selected as the probe set is shown in Fig. 4. The plot shows that they have comparable performance, with NMF better than SVD. Combining the two methods, *i.e.*, using the average output of these two systems as the final prediction would further improve the performance. Importantly, both collaborative filtering approaches perform significantly better than the naïve direct search.

6. PBC-CNN Models: an example of unsupervised hyper-training phase

The results of the last section validate our claim that model recommendation is able to select useful models. The other important issue remains how to generate a large collection of potentially "expressive" models. ESVMs are restricted in that each model is highly tuned for a specific instance, and thus constrained to one particular set of categories when building models. However, models informative across categories and datasets could be achieved via *unsupervised hyper-training*. In this section, we will give an example of hyper-training using predictable discriminative binary codes (PBCs) [44] modified to be estimated in an unsupervised fashion. We hyper-train on one dataset and test on another to demonstrate the effectiveness of our approach.

Each bit in PBC corresponds to a specific model, which can be viewed as a partition of the shared feature space of labeled categories across the low-density region induced by discrimination and learnability. Given unlabeled data, we obtain pseudo-labeled data by Max-Min sampling [11] and then cast the issue as semi-supervised PBCs [7]. Besides the unsupervised aspect, this procedure is also generalizable into web-scale image scenarios. As before, it is important to note that many other models could be created during hyper-training. We use this model only to validate our approach.

6.1. Model Generation via Unsupervised Hyper-Training

For the feature space, we use convolutional neural networks (CNNs) features [26] due to their demonstrated high performance, although other features could be used. In addition, consistent with recent work, we use category-independent region proposals as the basic processing and decision units. Given a large corpus of unlabeled training images, we first generate region proposals using selective search [52]. For each region proposal, we extract a 4,096-D feature vector $fc7$ from the final hidden layer of the pre-trained CNNs structure on ILSVRC 2012 [46] (without fine-tuning on other datasets) [26, 15, 18]. Now we have constructed a feature space with unlabeled proposals.

The current methods for producing PBC need label information [44, 7]. For our unsupervised discovery of PBC, we obtain several distinct and compact groups of labeled data by employing Max-Min sampling [11], which we use as initial pseudo-labeled data. We then produce the semi-supervised PBCs from these labeled datasets. Specifically, from a large proposal pool $\mathcal{P}$ we first draw a subset $\mathcal{A}_\mathcal{S}$ by random subsampling. Within $\mathcal{A}_\mathcal{S}$, we create prototype sets $\mathcal{B}_{\mathcal{PL}}$ by Max-Min sampling. We view $\mathcal{B}_{\mathcal{PL}}$ as pseudo-labeled data and the remaining ones in $\mathcal{A}_\mathcal{S}$ that are still unlabeled as $\mathcal{C}_{\mathcal{UL}}$. We use the same setup and parameters for the Max-Min sampling procedure reported in [11], resulting in $\mathcal{B}_{\mathcal{PL}}$ consisting of 30 categories with 6 samples per category. Based on only the pseudo-labeled data in $\mathcal{B}_{\mathcal{PL}}$, we learn a 10-D prototype PBC [44]. We select and add 50 samples to each pseudo-category from the unlabeled pro-

posals $\mathcal{C}_{\mathcal{UL}}$ using category specific attributes only to expand coverage as in [7]. Using this augmented dataset $\mathcal{D}_{\mathcal{AUG}}$, we retrain a new 10-D PBC, *i.e.*, 10 models. To ensure diversity, the subsampling procedure repeats for T times, and we learn $10T$ models in total. They thus build up our model library for widespread visual/attribute coverage.

6.2. Experimental Evaluation

To show that the PBC models generated by unsupervised hyper-training are informative across categories, we consider large-scale detection tasks across different datasets.

We use the entire PASCAL VOC 2007 dataset to generate model library and ratings store. Store tasks are generated similarly as the previous ESVM-based system. The final ratings store is of size $10{,}000 \times 10{,}000$. For the hidden tasks, we consider detection of 107 object categories on the SUN 09 test dataset [8], which span from regions (*e.g.*, road, sky) to well defined objects (*e.g.*, car, sofa) and highly deformable objects (*e.g.*, river, curtain). The input tasks are generated by randomly sampling 10 images from the corresponding category on the SUN 09 training dataset. For those categories whose numbers of training samples are smaller than 10, we use all the training samples. We also randomly generate 10 input tasks per hidden task, and obtain the average performance. For each PBC model, we follow the typical detection pipeline of R-CNN as in [15].

This is a challenging problem given that feature learning is implemented on ImageNet and PBC models are hyper-trained on PASCAL, while the system is finally tested on SUN. They are very different domains. The farther away from these source domains, the more pertinent the test dataset will be for experimental evaluation. (Note that testing on other datasets, *e.g.*, PASCAL VOC 2012 and ImageNet, cannot serve this purpose due to shared data.) Compared with PASCAL, on the SUN dataset the number of objects and the amount of noise significantly increase with far more annotated object categories and typically 7 object classes per image [8]. Additional contextual information is thus usually necessary to boost the detection performance [8, 9].

6.2.1 Ensemble Model Recommendation

To deal with the fact that different input tasks may require different factor numbers, we perform collaborative filtering using SVD and NMF with factor numbers d ranging from 100 to 1000. For each d, we evaluate their precision on the input task, and we then average the ratings across all these factor configurations. The final selected models are ranked according to this averaged prediction. Consistent with early work in consumer recommender system [55], we found that this method gives by far the best performance. After recommending the top-k desired models, we calibrate them by standard Platt scaling [42, 33] on the input task to obtain comparable scores and then perform majority voting as the fused score for each proposal, followed by non-maximum suppression. Moreover, following the standard bounding box regression procedure [15], we also learn additional bounding box regressors on the input tasks, and rectify the region proposals at test time to mitigate mislocalizations induced by proposal based object detection. The average ensemble recommendation result is shown in Fig. 5.

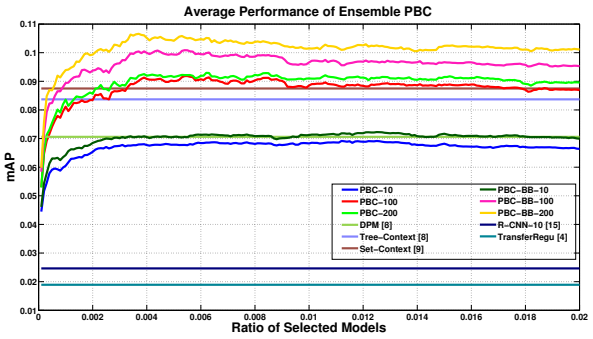

Figure 5. Average performance of ensemble PBC model recommendation with varied input task size over 107 categories on the SUN 09 dataset. X-axis: ratio of recommended models in the library. The ensemble model recommendation results for input tasks of size 10, 100, 200 *without* and *with* bounding box regression are shown in two sets of curves—PBC-10, PBC-100, PBC-200 and PBC-BB-10, PBC-BB-100, PBC-BB-200, respectively. We also show two sets of baselines to calibrate the results: the top three horizontal lines are the average performance of DPM using the entire training dataset and additionally annotated training data *without* and *with* introducing and encoding complex contextual relationships, respectively; the bottom two horizontal lines are the average performance of R-CNN directly trained and additionally transferred from our source models using 10 images from the input tasks, respectively. Note that accurate generic object detectors can be obtained based on input tasks that are from different categories and domains compared with where the PBC models are learned.

Baselines We compare against five strong baselines: 1) DPM using the entire SUN training dataset and 26,000 additionally annotated objects [8]; 2) Tree-structured graphical model [8] and 3) Set-based representation [9] to encode complex contextual relationships; 4) Direct R-CNN training using 10 images from the input tasks [15]; 5) Transfer regularization [4] with R-CNN and our source models.

Fig. 5 shows that ensemble of PBC models works consistently well as the size of the recommended model increases, as shown in the six curves. The result is quite significant if one notices the difference in numbers of training samples: Using only 10 images to select models generated from an out-of-domain dataset our approach is not only better than R-CNN directly trained from few samples, but it is also better than supervised DPM trained from *lots of* in-domain data (hundreds to thousands), comparable to DPM with additional contextual models. It is hard to make

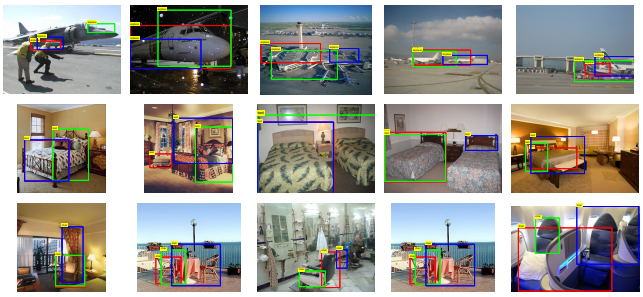

Figure 6. Collaborative detection of different models for the same task. For three representative tasks: airplane, bed, and chair (top to bottom), we show the top detection of the top-3 ranked models (red, green, and blue bounding boxes) on sample images (left to right). Note that they would detect the same, different parts of the same, or different objects. This complementary behavior explains the boosted performance of ensemble models in Fig. 5.

a direct comparison with Object Bank/Classemes since they use the output of all the models as new features to retrain a new classifier while we directly combine a subset of models. Experimentally, if we use Classemes features to retrain a detector with few samples, it is worse than R-CNN. For the SUN dataset, the mAP is low due to some hard categories. (The low mAP is of the same order of magnitude as in challenging datasets, *e.g.*, DPM on ImageNet.) Besides, in our case with large-scale sources of non-categorical classifiers, the conventional transfer learning techniques developed with well-trained categorical source classifiers [4] perform poorly due to induced negative transfer. This indicates that cross-source relations estimated by the recommender system are crucial to identify the relevant sources here. That is, because we have only few target samples and because the generic sources are weak, direct transfer will be very noisy.

Similar to the object distribution in a large-scale scenario, the performance distribution of models in the library for a specific task also follows a *power law*, which is implicitly encoded by the recommender system. Moreover, as we increase the sample size in the input task from 10 to 100 and 200, it shows steadily increased performance due to more accurate mAP, rank, calibration, and regressors estimation, and outperforms sophisticated contextual models. Bounding box regression provides additional performance boost as expected. This demonstrates that the recommender system both successfully builds expressive models during hyper-training and selects useful models based on input tasks.

6.2.2 Qualitative Visualization

Detection and Model Visualization To better understand the PBC models, we provide two types of representative visualization: 1) Collaborative detection of different models for the same task in Fig. 6; 2) Attributes-like behavior of the same model across different tasks in Fig. 7.

Continuous Category Space Discovery By calculating the number of shared models, a similarity matrix is obtained between the 107 categories. We use ForceAtlas [5, 17] to visualize it in Fig. 2. Although the models are generated without any label information, visually or functionally similar categories are naturally grouped together. It again shows the expressive power of the PBC models, which naturally identifies a continuous category space.

Figure 7. The same model is informative across different tasks. For five representative models (left to right), we show detection on sample images (top to bottom). Note that the models learned by unsupervised hyper-training can be to some extent interpreted as attribute detectors. For example, the first column corresponds to all staircase-like objects with vertical, horizontal, inclined, and curved orientations (bottom to top). This attributes-like behavior explains the generality of the models for new target tasks in Fig. 5.

7. Conclusions

We have shown the feasibility of generating new detectors for a new detection task given a large store of ratings of detectors on tasks. This enables generating detectors from a modest number of training samples in far less time than is needed by direct training. In addition, such a system could be made to improve continuously over time by adding new models and tasks to the ratings store. The specific implementation described in this paper is a first step. In particular, while we used certain factorization as our collaborative filtering technique, many different approaches have been developed in that community. It would be interesting to analyze which techniques are best adapted to different types of data. Similarly, we used one particular approach for fusing multiple (top-k) recommendations, but designing a good fusion strategy integrated with recommendation remains an important direction of research. Also, we have assumed a full ratings store matrix whereas, in general, one would envision a sparse matrix, especially in the regime of very large model libraries and task sets. Finally, our specific implementation in the context of detection tasks suggests a possible mechanism for a broader range of vision tasks.

Acknowledgments: This work was supported in part by U.S. Army Research Laboratory (ARL) under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016, and by an AWS in Education Coursework Grant.

References

[1] The infinite monkey theorem. http://en.wikipedia.org/wiki/Infinite_monkey_theorem.

[2] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6:1817–1853, 2005.

[3] Y. Aytar and A. Zisserman. Tabula rasa: Model transfer for object category detection. In *ICCV*, 2011.

[4] Y. Aytar and A. Zisserman. Enhancing exemplar svms using part level transfer regularization. In *BMVC*, 2012.

[5] M. Bastian, S. Heymann, and M. Jacomy. Gephi: an open source software for exploring and manipulating networks. In *International AAAI Conference on Weblogs and Social Media*, 2009.

[6] G. E. Box and N. R. Draper. *Empirical model-building and response surfaces*. John Wiley & Sons, 1987.

[7] J. Choi, M. Rastegari, A. Farhadi, and L. S. Davis. Adding unlabeled samples to categories by learned attributes. In *CVPR*, 2013.

[8] M. J. Choi, J. J. Lim, A. Torralba, and A. S. Willsky. Exploiting hierarchical context on a large database of object categories. In *CVPR*, 2010.

[9] R. G. Cinbis and S. Sclaroff. Contextual object detection using set-based classification. In *ECCV*, 2012.

[10] R. G. Cinbis, J. Verbeek, and C. Schmid. Segmentation driven object detection with fisher vectors. In *ICCV*, 2013.

[11] D. Dai and L. V. Gool. Ensemble projection for semi-supervised image classification. In *ICCV*, 2013.

[12] C. Doersch, A. Gupta, and A. A. Efros. Mid-level visual element discovery as discriminative mode seeking. In *NIPS*, 2013.

[13] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *IJCV*, 88(2):303–338, 2010.

[14] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 32(9):1627–1645, 2010.

[15] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[16] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. In *ECCV*, 2012.

[17] M. Jacomy, S. Heymann, T. Venturini, and M. Bastian. Forceatlas2, a continuous graph layout algorithm for handy network visualization. *Medialab center of research*, 560, 2011.

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, 2014.

[19] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural computation*, 6(2):181–214, 1994.

[20] M. Juneja, A. Vedaldi, C. Jawahar, and A. Zisserman. Blocks that shout: Distinctive parts for scene classification. In *CVPR*, 2013.

[21] J. B. Kadane and N. A. Lazar. Methods and criteria for model selection. *Journal of the American Statistical Association*, 99(465):279–290, 2004.

[22] O. Kallenberg. *Probabilistic symmetries and invariance principles*. Springer Science & Business Media, 2006.

[23] A. Karatzoglou, M. Weimer, and A. J. Smola. Collaborative filtering on a budget. In *AISTATS*, 2010.

[24] Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *TKDD*, 4(1):1–24, 2010.

[25] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37, 2009.

[26] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[27] I. Kuzborskij and F. Orabona. Stability and hypothesis transfer learning. In *ICML*, 2013.

[28] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *NIPS*, 2000.

[29] L.-J. Li, H. Su, E. P. Xing, and F.-F. Li. Object bank: A high-level image representation for scene classification & semantic feature sparsification. In *NIPS*, 2010.

[30] J. J. Lim, R. Salakhutdinov, and A. Torralba. Transfer learning by borrowing examples for multiclass object detection. In *NIPS*, 2011.

[31] B.-D. Liu, Y.-X. Wang, B. Shen, Y.-J. Zhang, and M. Hebert. Self-explanatory sparse representation for image classification. In *ECCV*, 2014.

[32] B.-D. Liu, Y.-X. Wang, Y.-J. Zhang, and B. Shen. Learning dictionary on manifolds for image classification. *PR*, 46(7):1879–1890, 2013.

[33] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

[34] P. Matikainen, R. Sukthankar, and M. Hebert. Classifier ensemble recommendation. In *ECCV Workshop on Web-scale Vision and Social Media*, 2012.

[35] P. Matikainen, R. Sukthankar, and M. Hebert. Model recommendation for action recognition. In *CVPR*, 2012.

[36] I. Misra, A. Shrivastava, and M. Hebert. Data-driven exemplar model selection. In *WACV*, 2014.

[37] I. Misra, A. Shrivastava, and M. Hebert. Watch and learn: Semi-supervised learning of object detectors from videos. In *CVPR*, 2015.

[38] M. Naphade, J. R. Smith, J. Tesic, S.-F. Chang, W. Hsu, L. Kennedy, A. Hauptmann, and J. Curtis. Large-scale concept ontology for multimedia. *IEEE MultiMedia*, 13(3):86–91, 2006.

[39] M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell. Zero-shot learning with semantic output codes. In *NIPS*, 2009.

[40] S. J. Pan and Q. Yang. A survey on transfer learning. *TKDE*, 22(10):1345–1359, 2010.

[41] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Bilinear classifiers for visual recognition. In *NIPS*, 2009.

[42] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*, 1999.

[43] A. Quattoni, M. Collins, and T. Darrell. Transfer learning for image classification with sparse prototype representations. In *CVPR*, 2008.

[44] M. Rastegari, A. Farhadi, and D. Forsyth. Attribute discovery via predictable discriminative binary codes. In *ECCV*, 2012.

[45] S. Ross, J. Zhou, Y. Yue, D. Dey, and J. A. Bagnell. Learning policies for contextual submodular prediction. In *ICML*, 2013.

[46] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. arXiv:1409.0575, 2014.

[47] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML*, 2007.

[48] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[49] T. Tommasi, F. Orabona, and B. Caputo. Learning categories from few examples with multi model knowledge transfer. *TPAMI*, 36(5):928–941, 2014.

[50] L. Torresani, M. Szummer, and A. Fitzgibbon. Efficient object category recognition using classemes. In *ECCV*, 2010.

[51] A. Tsymbal. The problem of concept drift: definitions and related work. Technical report, Computer Science Department, Trinity College Dublin, 2004.

[52] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *IJCV*, 104(2):154–171, 2013.

[53] Y.-X. Wang and Y.-J. Zhang. Nonnegative matrix factorization: A comprehensive review. *TKDE*, 25(6):1336–1353, 2013.

[54] M. Weimer, A. Karatzoglou, Q. V. Le, and A. Smola. Maximum margin matrix factorization for collaborative ranking. In *NIPS*, 2007.

[55] M. Wu. Collaborative filtering via ensembles of matrix factorizations. In *Proceedings of KDD Cup and Workshop*, 2007.

[56] F. Xiao and M. Hebert. Efficient model evaluation with bilinear separation model. In *WACV*, 2015.

[57] B. Yao, G. Bradski, and L. Fei-Fei. A codebook-free and annotation-free approach for fine-grained image categorization. In *CVPR*, 2012.

[58] Z.-H. Zhou. *Ensemble methods: foundations and algorithms*. CRC Press, 2012.