Learning Semantic Image Representations at a Large Scale

Yangqing Jia

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2014-93
http://www.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-93.html

May 16, 2014

Learning Semantic Image Representations at a Large Scale

by

Yangqing Jia

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Trevor Darrell, Chair
Professor Alexei Efros
Professor Thomas Griffiths

Spring 2014

Learning Semantic Image Representations at a Large Scale

Abstract

Learning Semantic Image Representations at a Large Scale

by

Yangqing Jia

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Trevor Darrell, Chair

I present my work towards learning a better computer vision system that learns and generalizes object categories better, and behaves in ways closer to what human behave. Specifically, I focus on two key components of such a system: learning better features, and revisiting existing problem statements. For the first component, I propose and analyze novel receptive field learning and dictionary learning methods, mathematically justified by the Nyström sampling theory, that learn more compact and effective features for object recognition tasks. For the second component, I propose to combine otherwise independently developed computer vision and cognitive science studies, and present the first large-scale system that allows computers to learn and generalize closer to what a human learner will do. I also provide a large-scale human behavior database, which will hopefully enable further research along this research direction.

Following the recent success of convolutional neural networks, I present and release a well-engineered framework for general deep learning research, and provide an extensive analysis on the generality of deep features learned from the state-of-the-art CNN pipeline: whether they serve as a general-purpose visual descriptor that could be adopted in various applications, and future research directions made possible by such general features.

To Sizhu, and my parents.

Contents

Acknowledgments

It is a significant stage of life to spend five years working on a PhD degree, and I am very grateful to have worked with wonderful people during this period. First and foremost, I would like to thank my advisor Trevor Darrell, for being a great mentor to introduce me into the wonderful field of academic research and to encourage me to explore new fields and research directions. I would also like to thank Tom Griffiths, Jitendra Malik, Alyosha Efros, and Bruno Olshausen, for high-level and interdisciplinary guidances allowing me to see beyond my otherwise limited research field.

I had never imagined a more enjoyable graduate school life before I joined Berkeley, and it is an honor to meet and work with fellow postdocs and graduate students: Oriol Vinyals, Jon Barron, Sergey Karayev, Trevor Owens, Hyun Oh Song, Ning Zhang, Judy Hoffman, Allie Janoch, Jon Long, Jeff Donahue, Evan Shelhamer, Joshua Abbott, Joseph Austerweil, Dave Golland, Matthieu Salzmann, Brian Kulis, Mario Fritz, Mario Christoudias, Ross Girshick, Sergio Guadarrama, and many others. Grad school has been unimaginably colorful with your company.

I appreciate my internship days at the NEC Labs America and Google Research. My thanks go to Chang Huang, Kai Yu, Mei Han, Thomas Leung, Alexander Toshev, and Sergey Ioffe, for offering the great opportunity for me to enlarge my vision, and to boldly go into the era of large-scale deep learning.

Last but not least, I am deeply indebted to the love, tolerance and support from my wife Sizhu and my parents. This thesis is dedicated to them with my sincere gratitude.

Chapter 1

Introduction

A fundamental problem in computer vision is *object recognition*: given an image composed of a grid of raw pixel values, one needs to design a computer system that identifies the objects present in this image. It is known that humans are particularly good at such problems, being able to learn quickly from very few examples (with the help of life-long visual experiences), and to adapt to various visual input conditions like illumination, rotation and deformation. By its nature, computer vision has been a vague problem, requiring one to design computer vision algorithms as well as evaluation criteria to achieve human-like vision systems.

Two key trends have driven the vision field forward during the recent years. With the highly structured visual input, it is always a challenge to find visual features that preserve useful information and provide satisfying invariance against variations. Breakthroughs in vision applications often comes with more powerful features, such as SIFT [78], HOG [25], and the recent rediscovery of convolutional neural network (CNN) features [64, 65]. At the same time, defining more precise problem statements as well as benchmarks almost always provides new perspectives and directions to the research field. This both helps better understanding of existing systems, and enables more powerful systems to be learned from ever-growing data.

In this thesis I present work that aligns with such trends: to learn a better computer vision system that learns and generalizes object categories better, and behaves in ways closer to what human learners do. As any attempt towards such a system would involve a number of key problems and challenges, I will introduce and discuss my contribution towards two problems in such vision systems: to learn better image features with solid theoretical justifications, and to re-visit the existing object recognition problem statement, proposing a novel, cognitive science inspired system that learns and generalizes object categories similar to human learners.

It is noteworthy that vision algorithms often call for efforts from the computer systems side, which enables one to learn from large-scale data and to learn complicated models. Such need is highlighted in the recent comeback of "deep learning", which employs the conventional wisdom of multi-layer, convolutional neural networks, but is usually trained with terabytes of data and millions of parameters. It is arguable that this could not be achieved

by novel computer architectures - distributed systems employing thousands of machines, and heterogeneous computing platforms such as Graphical Processing Units (GPUs). However, little systematic efforts have been made to provide a state-of-the-art codebase for the recent advances in vision and deep learning. In this thesis, I will also propose and provide an open-source library called "Caffe" for such needs, highlighting key design choices that make it efficient. By the time of this thesis, Caffe has gained much interest both in academia and industry, and has been supporting multiple research projects both inside and outside Berkeley.

Due to the scale of topics involved in this thesis, I will leave the background and literature reviews to each individual chapter, which will be a self-containing part with discussion on how it fits in the overall theme of this thesis. Here I briefly summarize the main contribution of this thesis:

- To better understand the nature of image feature learning by presenting both theoretical and empirical analysis towards more compact and effective image features, showing improvement on state-of-the-art image classification tasks (Chapter 2 and 3).
- To connect the gap between "laboratory style" object categorization and concept learning problems that are closer to human cognitive behavior, pushing the frontier on both machine vision and cognitive science (Chapter 4 and 5).
- To present a well-engineered, most-efficient open-source framework that fosters future computer vision and machine learning research, with systematic analyses of state-of-the-art deep learning approaches (Chapter 6).

Earlier versions of this work have been presented in smaller parts over the course of several research papers [56, 57, 58, 55, 31]. This dissertation goes through, and shows how individual components contribute to the overall contribution as follows:

Chapter 2 focuses on finding better image feature representations, which is the fundamental part of all recognition tasks. Specifically, we focus on the building block of state-of-the-art feature learning pipelines: a two-stage pipeline containing a local encoding stage and a spatial pooling stage. We show that an over-complete pooling receptive field design, combined with a discriminative feature selection scheme, is able to capture richer between-class variance and achieve state-of-the-art performance on benchmark datasets. While this chapter only focuses on networks with only a single coding and pooling stage, the algorithm may be extended to deeper, multi-stage networks, where one may construct a criterion for feature selection by examining the gradients of upstream networks.

Chapter 3 then gives a theoretical justification of over-complete features and greedy feature selection. One could view the feature selection as a sampling problem from a potentially infinite-dimensional feature space, whose behavior could be well understood by the covariance matrix between features. While the Nyström sampling theory has been well studied from a purely machine learning perspective, not much use has been proposed beyond simple methods such as K-means and SVMs. This chapter will show a natural connection

between feature selection and Nyström sampling, justifying the use of simple, greedy feature selection schemes discussed earlier in the chapter.

Having discussed the feature learning algorithms, **Chapter 4** moves on to a higher level and analyzes the question of visual concept learning, originating from psychology and cognitive science. Specifically, we address the gap between the behavior of human and that of machines on learning a novel category by combining knowledge from two distinctive fields - machine vision and cognitive science - that have developed separately in the previous decades. As the scale of our problem has never been tried in either fields, I propose and collect a systematic testing scheme, and present the first system that is capable of learning novel concepts directly from perceptual inputs, in a much larger scale than existing cognitive science approaches usually address.

Chapter 5 employs the visual concept learning framework, and presents the solution to a more conventional machine vision problem: to enable an agent that is able to learn from a large number of object categories, but is also capable of adapting to different task scenarios, and only predicting object categories that are semantically related to the current task context. The chapter benefits from the cognitive science model presented in Chapter 3, and to the best of my knowledge is the first machine vision system that addresses the semantic difference during training and testing time.

Last but not least, I present **Chapter 6** in a more exploratory fashion than previous chapters, by evidencing and analyzing the emergence of object-level information along the multiple stages of a very deep convolutional neural network, as well as the applicability of deep features as a general-purpose feature that effectively replaces SIFT and HOG in state-of-the-art vision tasks, based on the Caffe framework that I developed and released. Chapter 6 also discusses key design choices of Caffe that plays as the backbone of all algorithms presented in the chapter.

Chapter 2

Receptive Field Learning for Image Features

A key component in the object recognition pipeline is extracting robust yet representative features from perceptual inputs, usually in the format of raw pixels. Such features should be able to further support high-level interpretations such as categorization and detection, and the vision community has converged to specific architectures for feature extraction in the recent decade. Most notably, such architectures use a convolutional approach that encodes local image patches and spatially pools the output, and then stacks such convolutional components in a multi-layer fashion to build mid and high level features. Despite various ways in which such networks could be constructed (e.g. with handcrafted features or fully trained), such structures have remained effective in various applications, including digit recognition [74], object detection [24], object classification [117], and the recent success of convolutional neural networks in large-scale classification tasks [65].

This chapter focuses on the building block of such approaches - a single-layer network that contain one local coding stage and one spatial pooling stage. Specifically, we proposes a novel approach to perform pooling to obtain more selective features for object recognition, achieving higher performance on benchmark datasets than conventional pooling approaches do. We then explain the theoretical justification of a common phenomenon found in the single-layer network analysis: higher dimensional features almost always lead to better classification performance. This chapter focuses on the single-layer network for clarity, but the results we found would apply to multi-layer networks as well.

2.1 Background

Over-completely encoded features have been shown to provide state-of-the-art performance on various applications, see *e.g.,* [85, 77, 120, 19]. In computer vision, locally encoded and spatially pooled feature extraction pipelines work particularly well for image classification. Such pipelines usually start from densely extracted local image patches (either normalized

raw pixel values or hand-crafted descriptors such as SIFT or HOG), and perform dictionary learning to obtain a dictionary of codes (also called filters). The patches are then encoded into an over-complete representation using various algorithms such as sparse coding [85, 112] or simple inner product with a non-linear post-processing [20, 65]. After encoding, spatial pooling with average or max operations are carried out to form a global image representation [117, 13]. The encoding and pooling pipeline may be stacked in a deep structure to produce a final feature vector, which is then used to predict the labels for the images usually via a linear classifier or a densely connected multilayer neural network.

During the last decade, much emphasis has been directed at the coding step. Dictionary learning algorithms have been discussed to find a set of basis that reconstructs local image patches or descriptors well [80, 20], and several encoding methods have been proposed to map the original data to a high-dimensional space that emphasizes certain properties, such as sparsity [85, 117, 118] or locality [112]. Recent papers [19, 93, 20] have explored the relationship between dictionary learning and encoding, and have proposed simple yet effective approaches that achieve competitive results. The neuroscience justification of coding comes from simple neurons in the human visual cortex V1, which have been believed to produce sparse and over-complete activations [85].

Similarly, the idea of spatial pooling dates back to Hubel's seminal paper about complex cells in the mammalian visual cortex [51], which identifies mid-level image features that are invariant to small spatial shifting. The spatial invariance property also reflects the concept of locally orderless images [63], which suggests that low-level features are grouped spatially to provide information about the overall semantics. Most recent research on spatial pooling aims to find a good pooling operator, which could be seen as a function that produces informative statistics based on local features in a specific spatial area. For example, average and max pooling strategies have been found in various algorithms respectively, and systematic comparisons between such pooling strategies have been presented and discussed in [13, 11]. Recently, Coates et al. proposed to pool over multiple features in the context of deep learning [21].

However, relatively little effort has been put into better designs or learning of better spatial regions for pooling, although it has been discussed in the context of learning local descriptors [114]. A predominant approach to define the spatial regions for pooling, which we will also call the receptive fields (borrowing the terminology from neuroscience) for the pooled features, comes from the idea of spatial pyramids [70, 117], where regular grids of increasing granularity are used to pool local features. The spatial pyramids provide a reasonable cover over the image space with scale information, and most existing classification methods either use them directly, or use slightly modified/simplified versions.

In addition, recent research has revealed a particularly interesting finding [19, 93, 20, 98] that very simple patch-based algorithms like K-means or even random selection, combined with feed-forward encoding methods with a naive nonlinearity, produces state-of-the-art performance on various datasets. Explanation of such phenomena often focuses on the local image patch statistics, such as the frequency selectivity of random samples [98], but does not offer an asymptotic theory on the dictionary learning behavior. We will show later in

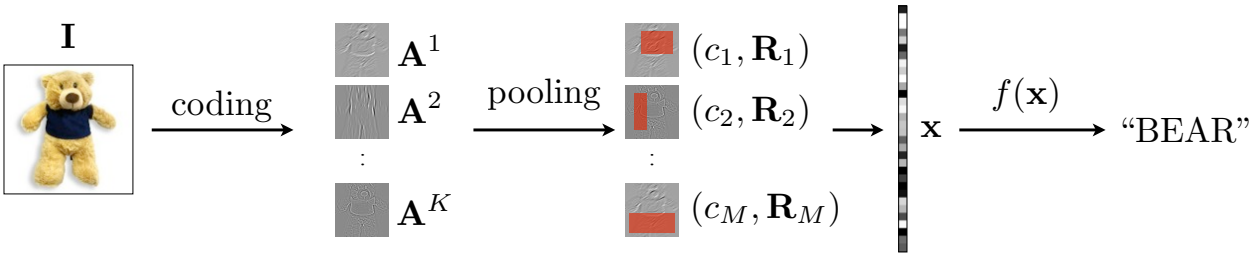


Figure 2.1: The image classification pipeline. See Section 2.2 for details.

the chapter that a Nyström sampling based interpretation explains such phenomenon well by providing asymptotic bounds to the observed accuracy, and that such interpretation will lead to an efficient, unsupervised feature selection paradigm.

2.2 The Classification Pipeline

Before the introduction of the proposed methods, we briefly review the image classification pipeline we adopted, which leads to the problem of learning the receptive fields for spatial pooling. Specifically, we will focus on two-layer classification approaches.

We illustrate the pipeline from raw images to the prediction of class labels in Figure 2.1. Specifically, starting with an input image $\mathbf{I}$, two stages are usually adopted to generate the global feature, as we formally define below.

(1) Coding. In the coding step, we extract local image patches, and encode each patch to K activation values based on a codebook of size K (learned via a separate dictionary learning step). These activations are typically binary (in the case of vector quantization) or continuous (in the case of e.g. sparse coding). It is generally believed that having an over-complete ($K \gg$ the dimension of patches) codebook while keeping the activations sparse helps classification, especially when linear classifiers are used in the later steps.

Recently, Coates et al. [20] have shown that relatively simple dictionary learning and encoding approaches lead to surprisingly good performances. To learn a dictionary $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_K]$ of size K from randomly sampled patches $\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_N\}$ each reshaped as a vector of pixel values, two simple yet effective approaches are advocated:

1. K-means, which minimizes the squared distance between each patch and its nearest code: $\min_{\mathbf{D}} \sum_{i=1}^{N} \min_j \|\mathbf{p}_i - \mathbf{d}_j\|_2^2$.

2. OMP-M, which learns a dictionary that minimizes the reconstruction error, with the constraint that each patch is modeled by a linear combination of at most M codes: $\min_{\mathbf{D},\boldsymbol{\alpha}_i} \sum_{i=1}^{N} \|\mathbf{p}_i - \mathbf{D}\boldsymbol{\alpha}_i\|_2^2$, where the length of each dictionary entry $\mathbf{d}_j$ is 1, and the cardinality of each reconstruction coefficient $\boldsymbol{\alpha}_i$ is at most M.

For encoding, Coates et al. also propose to substitute sparse coding by the following efficient approaches:

1. Triangle coding [19], which computes the activation of code k for a patch $\mathbf{p}$ as $f_k(\mathbf{x}) = \max\{0, \mu(\mathbf{z}) - z_k\}$, where z_k is the distance from $\mathbf{p}$ to the k-th code $\mathbf{d}_k$, and $\mu(\mathbf{z})$ is the mean of distances from $\mathbf{p}$ to all codes.

2. Soft thresholding, which computes the inner product between $\mathbf{p}$ and each code, with a fixed threshold parameter α: $f_k(\mathbf{x}) = \max\{0, \mathbf{d}_k^\top \mathbf{p} - \alpha\}$

We refer to [20] for a systematic discussion about different dictionary learning and encoding algorithms. In our experiment, we will adopt these standard approaches in order to isolate the contribution of spatial pooling from the choice of different coding methods. Since local patches are usually extracted densely in a grid-based fashion, we will organize the activations of image $\mathbf{I}$ as a set of matrices denoted by $\{\mathbf{A}^1(\mathbf{I})\mathbf{A}^2(\mathbf{I}), \cdots, \mathbf{A}^K(\mathbf{I})\}$, one for each code in the codebook, whose element $A^k_{ij}(\mathbf{I})$ contains the activation of code $\mathbf{d}_k$ for the local image patch at spatial location (i, j).

(2) Pooling. Since the coding result are highly over-complete and highly redundant, the pooling layer aggregates the activations over certain spatial regions of the image to obtain an M dimensional vector $\mathbf{x}$ as the global representation of the image. Each dimension of the pooled feature $\mathbf{x}_i$ is obtained by taking the activations of one code in a specific spatial region (shown as the red rectangular in Figure 2.1), and performing a predefined operator (usually average or max) on the set of activations.

We follow a similar approach to that in [12] to formally define pooled features. Specifically, given an operator op that maps a set of real values to a single real value (e.g. by taking their average), a pooled feature x_i can be defined based on the selection of a code indexed by c_i and a spatial region denoted by $\mathbf{R}_i$:

$$x_i = \text{op}(\mathbf{A}^{c_i}_{\mathbf{R}_i}) \tag{2.1}$$

Borrowing the definition from neuroscience, we call $\mathbf{R}_i$ the *receptive field* for the pooled feature, which could be seen as a binary mask over the image. $\mathbf{A}^{c_i}_{\mathbf{R}_i}$ is then the set of activations of code c_i in the receptive field $\mathbf{R}_i$.

This definition provides a general definition that embraces existing pooling algorithms. For example, commonly used operators involve computing the statistics of the activations under the p-norm:

$$x_i = \frac{1}{|\mathbf{R}_i|}\left(\sum\nolimits_{\alpha_i \in \mathbf{A}^{c_i}_{\mathbf{R}_i}} \alpha_i^p\right)^{\frac{1}{p}} \tag{2.2}$$

when $p = 1$ this corresponds to the average pooling, and when $p \to \infty$ this corresponds to the max pooling.

We focus on the definition of receptive fields for pooling. The simplest form of pooling takes the whole image as the receptive field, thus assuming a bag-of-words model where

spatial information is ignored. The more commonly adopted spatial pooling approach [70, 117] pools features from multiple levels of regular grids, thus defining a pyramid of pooled features. Given a set of K codes and a set of N receptive fields, the pooled features are then defined by taking the Cartesian product of the codes and the receptive fields, yielding a KN-dimensional global feature.

Finally, a classifier, usually linear SVM or logistic regression, is trained using the global feature vector to predict the final label of the image as $y = f(\mathbf{x}; \boldsymbol{\theta})$.

2.3 Receptive Field Learning for Pooled Image Features

While significant efforts have been placed on the coding part of the classification pipeline, the pooling step has received relatively little attention. Existing research on pooling mainly focuses on the analysis of the pooling operator, such as in [11]. Specifically, spatial regions are almost always defined on regular grids [117], which may not guarantee to be optimal. As a simple example, to distinguish most indoor and outdoor scenes, a human may look for the existence of the horizon, which could be captured by thin horizontal pooling regions over the image. Spatial grids, even with a pyramid structure, fail to provide such information. Such receptive fields may be dataset-dependent, leading us to ask the question *"are spatial pyramids optimal for image classification?"*, the answer to which is often neglected by existing algorithms.

Instead of arbitrarily defining heuristic receptive fields, we aim to explicitly learn the receptive fields for classification tasks. Specifically, we propose to adaptively learn such regions by considering the receptive fields additional parameters, and jointly learning these parameters with the subsequent classifiers. The resulting benefit is two-fold: receptive fields tailored to classification tasks increase the overall accuracy of classification; in addition, with the help of such mid-level features, we are able to use a much lower-dimensional feature to achieve the state-of-the-art performance. We experiment with our algorithm on the benchmark CIFAR-10 dataset and other datasets, and report a significant improvement in both accuracy and efficiency.

Inspired by the selectivity of complex cells in the visual cortex, we propose to learn the pooled features adaptively. Specifically, learning a set of M pooled features is equivalent to learning the parameters $\mathcal{C} = \{c_1, c_2, \cdots, c_M\}$ and $\mathcal{R} = \{\mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_M\}$ [1]. To this end, we note that the pooled features are directly fed into the final classifier, and propose to jointly learn the classifier parameters $\boldsymbol{\theta}$ together with the pooling parameters. Thus, given a set of training data $\mathcal{X} = \{(\mathbf{I}_n, \mathbf{y}_n)\}_{n=1}^N$, the joint learning leads to solving the following

[1] For simplicity, we will use the max operator, but note that any operator could also be incorporated in our framework.

(a) (b) (c)

Figure 2.2: An example of over-complete rectangular bins based on a 4×4 super-pixel setting: (a) super-pixels; (b) spatial pyramid bins; (c) over-complete rectangular bins.

optimization problem:

$$\begin{aligned} \min_{\mathcal{C},\mathcal{R},\boldsymbol{\theta}} \quad & \frac{1}{N}\sum_{n=1}^{N}\mathcal{L}(f(\mathbf{x}_n;\boldsymbol{\theta}),\mathbf{y}_n) + \lambda\,\mathrm{Reg}(\boldsymbol{\theta}) \\ \text{where} \quad & x_{ni} = \mathrm{op}(\mathbf{A}^{c_i}_{n,\mathbf{R}_i}) \end{aligned} \tag{2.3}$$

where we assume that the coding from $\mathbf{I}_n$ to $\{\mathbf{A}^{c_i}_n\}_{i=1}^K$ is done in an unsupervised fashion, as has been suggested by several papers such as [19]. We call this method receptive field learning, as the receptive fields are learned in such a way that information most relevant to the classification task will be extracted.

One practical issue is that solving the optimization problem (2.3) may be impractical, as there is an exponential number of receptive field candidates, leading to a combinatorial problem. Numerical solutions are also difficult, as the gradient with respect to the pooling parameters is not well-defined. Thus, instead of searching in the space of all possible receptive fields, we adopt the idea of over-completeness in the sparse coding community. Specifically, we start from a set of reasonably over-complete set of potential receptive fields, and then find a sparse subset of such pooled features. The over-completeness enables us to maintain performance, while the sparsity allows us to still carry out classification efficiently during testing time.

2.3.1 Over-complete Receptive Fields

The exponential number of possible receptive fields arises when we consider the inclusion and exclusion of single pixels individually. In practice this is often unnecessary, as we expect the active pixels in a receptive field to be spatially contiguous. In this work, we use receptive fields consisting of rectangular regions[2]: this provides us a reasonable level of over-completeness,

[2]As a side note, we also experimented with receptive fields that are sampled from an Ising model on the fly during training, but rectangular regions worked empirically better, possibly because the additional

as there are $O(n^4)$ different rectangular receptive fields for an image containing $n \times n$ pixels. In addition, since the motivation of spatial pooling is to provide tolerance to small spatial displacements, we build the rectangular regions upon super-pixels, which are defined as dense regular grids on the image. Figure 2.2 shows an example of such rectangular receptive fields compared with regions defined by the spatial pyramid on a 4×4 grid.

Given the set of P over-complete regions, which we denote by $\mathcal{R} = \{\mathbf{R}_1, \mathbf{R}_2, \cdots, \mathbf{R}_P\}$, and the dictionary $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \cdots, \mathbf{d}_K\}$ of size K, we can define a set of PK potential pooled features based the Cartesian product $\mathcal{R} \times \mathcal{D}$. Specifically, the i-th receptive field and the j-th code jointly defines the $(K \times i + j)$-th pooled feature as $x_{K \times i + j} = \text{op}(\mathbf{A}^j_{\mathbf{R}_i})$. Note that when the coding and pooling are both carried out in an over-complete fashion, the resulting pooled feature is usually very high-dimensional.

2.3.2 Structured Sparsity for Receptive Field Learning

While it is possible to train a linear classifier using the high-dimensional pooled feature $\mathbf{x}$ above, in practice it is usually beneficial to build a classifier using relatively low-dimensional features. In addition, for multiple-label classification, we want the classifiers of different labels to share features. This brings two potential advantages: feature computation could be minimized, and sharing features among different classifiers is known to provide robustness to the learned classifiers. To this end, we adopt the idea of structured sparsity [88, 99], and train a multiple-class linear classifier $\mathbf{y} = f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$ via the following optimization problem:

$$\min_{\mathbf{W},\mathbf{b}} \quad \frac{1}{N}\sum_{n=1}^{N} l(\mathbf{W}^\top \mathbf{x}_n + \mathbf{b}, \mathbf{y}_n) + \frac{\lambda_1}{1}\|\mathbf{W}\|^2_{\text{Fro}} + \lambda_2 \|\mathbf{W}\|_{1,\infty} \tag{2.4}$$

where $\mathbf{y}_i$ is the L-dimensional label vector coded in a $1 - of - L$ fashion, with values taken from $\{-1, +1\}$ given L classes. $\mathbf{x}_i$ is an M-dimensional feature vector defined by over-complete pooling in the previous subsection, and $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_L]$ is a $M \times L$ weight matrix containing the weight vector for the L classifiers.

Two regularization terms are adopted in the optimization. The squared Frobenius norm $\|\mathbf{W}\|^2_{\text{Fro}}$ aims to minimize the structured loss in the classical SVM fashion, and the second regularizer is the $L_{1,\infty}$ norm of the matrix $\mathbf{W}$:

$$\|\mathbf{W}\|_{1,\infty} = \sum_{i=1}^{M} \|\mathbf{W}_{i,\cdot}\|_\infty = \sum_{i=1}^{M} \max_{j \in \{1,\cdots,L\}} |W_{ij}| \tag{2.5}$$

where $\mathbf{W}_{i,\cdot}$ denotes the i-th row of the matrix W. This regularizer introduces structured sparsity by encouraging the weight matrix $\mathbf{W}$ to be row-wise sparse, so that the classifiers for different classes tend to agree on whether to use a specific feature, and when combined together, only jointly use a subset of the over-complete pooled features. The addition of

flexibility of Ising models leads to over-fitting the training data, and the spatial inconsistency may render randomly sampled receptive fields not as useful in classification tasks.

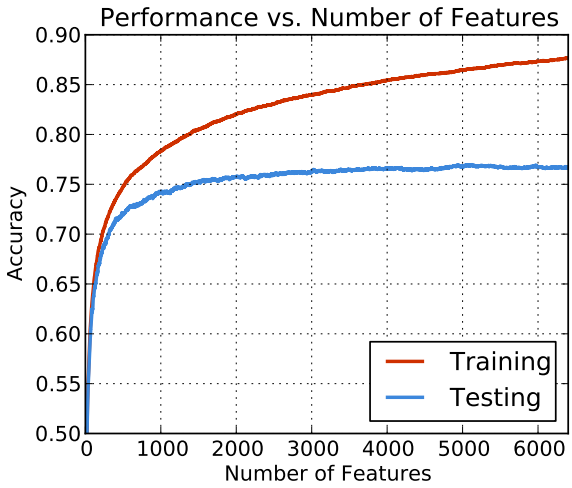


Figure 2.3: Performance vs. number of selected features, with the experiment setting in Table 2.1 of Section 2.5.

the $L_{1,\infty}$ norm also provides a elastic-net like regularization, which is known to perform well when the dimension of data is much higher than the number of data points [126].

For optimization considerations, we use the multi-class extension of the binomial negative log likelihood (BNLL) loss function [87]:

$$l(\mathbf{W}^\top \mathbf{x} + \mathbf{b}, \mathbf{y}) = \sum_{i=1}^{L} \ln(1 + e^{-\mathbf{y}_i(\mathbf{W}_{\cdot,i}^\top \mathbf{x} + b_i)}) \tag{2.6}$$

The choice of the BNLL loss function over the hinge loss is mainly for computational simplicity, as the gradient is easier to compute for any input. In practice, the performance does not change much if we use the hinge loss instead.

2.4 Fast Approximate Learning with Feature Grafting

Jointly optimizing (2.4) is still a computationally challenging task despite its convexity, due to the over-completeness in both coding and pooling. While it is possible to carry out the computation on smaller-scale problems like Caltech-101, we adopt a greedy approach to train the model for larger-scale problems. Inspired by the matching pursuit algorithm in dictionary training and the grafting algorithm [87] in machine learning, we start with an empty set of selected features, incrementally add features to the set, and retrain the model when new features are added.

Mathematically, we maintain a set $\mathcal{S}$ recording the set of currently selected features. At each iteration, for each feature index j that has not been not selected, we compute the score of the feature as the 2-norm of the gradient of the objective function (2.4), denoted by

$\mathcal{L}(\mathbf{W}, \mathbf{b})$, with respect to the corresponding weight vectors:

$$\text{score}(j) = \left\| \frac{\partial \mathcal{L}(\mathbf{W}, \mathbf{b})}{\partial \mathbf{W}_{j,\cdot}} \right\|_{\text{Fro}}^2 \tag{2.7}$$

We then select the feature with the largest score, and add it to the selected set $\mathcal{S}$. The model is retrained using the previously learned optimum solution as the starting point. From a boosting perspective, this can be considered as incrementally learning weak classifiers, but our method differs from boosting in the sense that the weights for already selected features are also updated when new features are selected.

As the speed of retraining drops when more features are added, we adopt an approximate retraining strategy: for each iteration t, we select an active subset $\mathcal{S}_A$ of $\mathcal{S}$ based on the score above. We then retrain the model with respect to the active set and the bias term only:

$$\mathbf{W}_{\mathcal{S}_A,\cdot}^{(t+1)}, \mathbf{b} = \arg\min_{\mathbf{W}_{\mathcal{S}_A,\cdot}, \mathbf{b}} \mathcal{L}(\mathbf{W}, \mathbf{b}) \tag{2.8}$$

with the constraint that $\mathbf{W}_{\bar{\mathcal{S}}_A,\cdot}$ keep unchanged. The intuition is that with an already trained classifier from the previous iteration, adding one dimension will only introduce small changes to the existing weights.

In practice, we found the performance of this approximate algorithm with the active set size less than 100 to be very close to the full retraining algorithm with a significant increase in computation speed. Figure 2.3 shows typical curves of the training and testing accuracy with respect to the number of iterations. The performance usually stabilizes with a significantly smaller number of features, showing the effectiveness of introducing structured sparsity into classifier learning.

2.5 Experiments

We will mainly report the performance of our algorithm on the CIFAR-10 dataset[3], which contains 50,000 32×32 images from 10 categories as training data, and 10,000 images as testing data.

We fix the dictionary learning algorithms to k-means clustering and the coding algorithms to triangular coding as proposed in [19] for CFAR-10. Such a coding strategy has been shown to be particularly effective in spite of its simplicity. We also tested alternative dictionary learning and coding algorithms, which led to similar conclusions. As our main focus is on learning receptive fields for pooled features, the results of different coding algorithms are omitted, and we refer to [20] for a detailed discussion about dictionary learning and coding algorithms.

For classification, when we use pre-defined receptive fields such as spatial pyramids, the SVM regularization term is chosen via 5-fold cross validation on the training data. When we

[3]http://www.cs.toronto.edu/ kriz/cifar.html

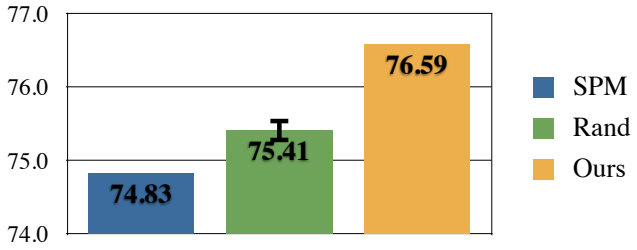


Figure 2.4: Performance comparison among spatial pyramid pooling, random feature selection and our method, all using the same number of features for the final classification. It can be observed that a few selected features could already achieve a comparatively high performance.

perform feature selection, we fix $\lambda_1 = 0.01$ (which is the best value when performing 5-fold cross validation for max pooling on a 2×2 regular grid) and drop λ_2, since the incremental feature selection already serves as a greedy approximation of the sparse constraint. Although the parameters are not tuned specifically for each configuration, we found it to perform well empirically under various scenarios.

2.5.1 Spatial Pyramid Revisited

It is interesting to empirically evaluate the performance of spatial pyramid regions against other choices of receptive fields. To this end, we trained a dictionary of size 200 (for speed considerations), and tested the performance of 3-layer spatial pyramid pooling against two algorithms based on over-complete receptive fields: (1) random selection from the over-complete pooled features, and (2) our method, both selecting the same number of features that spatial pyramid pooling uses. Results are shown in Figure 2.4. Our method outperforms SPM, but a more interesting finding is that the predefined spatial pyramid regions perform consistently worse than random selection, indicating that arbitrarily defined pooled features may not capture the statistics of real-world data well. With explicit learning of the pooling parameters, we achieved the highest performance among the three algorithms, showing the effectiveness and necessity of learning adaptive receptive fields.

2.5.2 The Effect of Spatial Over-completeness

One may ask if the performance increase could be obtained without over-completeness by simply using a denser grid. To answer this question, we examined the performance of our algorithm against the 2×2 pooling grid (which is used in [20] to obtain very high performance) and a denser 4×4 grid, associated with either average or max pooling. We also compared our method against random feature selection from the same pooling candidates. Table 2.1 summarizes the testing accuracy under various experimental settings, using a codebook size of 200.

Pooling Area	Method	Features	Accuracy
2×2	Ave	800	70.24
4×4	Ave	3,200	72.24
2×2	Max	800	66.31
4×4	Max	3,200	73.03
3-layer SPM	Max	4,200	74.83
OC + feat select	Max	800	73.42
		3,200	76.28
		4,200	76.59
		6,400	**76.72**
OC, all features	Max	20,000	76.44
OC + rand select	Max	800	69.48
OC + rand select	Max	3,200	74.42
OC + rand select	Max	4,200	75.41

Table 2.1: Comparison of different pre-defined pooling strategies and our method (over-complete (OC) + feature selection). Random selection from the same over-complete pooled features is also listed, showing the necessity of better receptive field learning.

Results from Table 2.1 demonstrates that denser pooling does help performance. The 4×4 grid increases the performance by about 3 percent compared to 2×2 pooling. However, with over-complete receptive fields we can almost always increase performance further. We achieved an 76.72% accuracy with only 200 codes, already close with state-of-the-art algorithms using much larger codebook sizes (Table 2.2). It is also worth pointing out that even random feature selection gives us comparable or better performance when compared to pre-defined pooling grids under the same number of feature dimension (e.g. compare the performance between 4×4 max pooling and randomly selecting $3,200$ features from an over-complete set of pooled features).

Further, the importance of feature selection lies in two aspects: first, simply using all the features is not practical during testing time, as the dimension can easily go to hundreds of thousands when we increase the codebook size. Feature selection is able to get very close performance compared to using all the features, but with a significantly lower dimensionality, which is essential in many practical scenarios. Usually, feature selection enables us to achieve a high performance with only a few features (Figure 2.3). Adding remaining features will only contribute negligibly to the overall performance. Second, performing feature selection has the potential benefit of removing redundancy, thus increasing the generalization ability of the learned classifiers [87, 108]. In our experiment in Table 2.1, the best performance is achieved with a few thousands features. Similarly, we found that with larger codebook sizes, using all the over-complete pooled features actually decreases performance, arguably due to the decrease of the generalization ability.

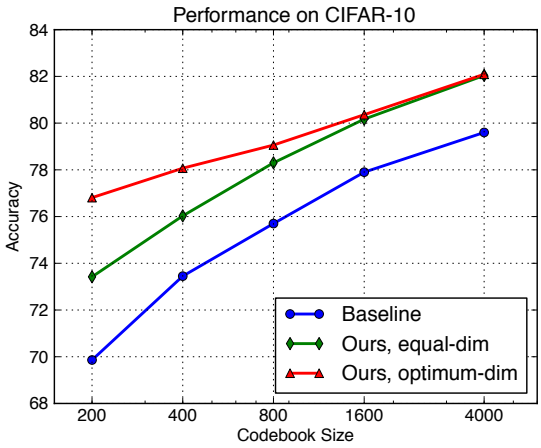


Figure 2.5: Testing accuracies on CIFAR-10 with and without over-complete pooling. In the figure, "equal-dim" selects the same number of features as the baseline (Coates et al.[19]), and "optimum-dim" selects the optimum number of features determined by cross-validation. (X-axis in log scale)

2.5.3 Larger Codebook vs. Better Spatial Pooling

Under the two-stage pipeline adopted in this work, there are effectively two possible directions to increase the performance: to increase the codebook size and to increase the pooling over-completeness. We argue that these two directions are complementary: the performance gain from our effort on pooling could not simply be replaced by increasing the codebook size, at least not easily. More importantly, as the codebook size grows larger, it becomes more difficult to obtain further performance gain, while it is still relatively easy to obtain gains from better pooling.

To empirically justify this argument, we trained multiple codebooks of different sizes, and compared the resulting accuracies with and without over-complete pooling in Figure 2.5. As can be observed, it becomes harder to obtain further performance gain by increasing the codebook size when we already have a large codebook, while using a better pooling strategy always brings additional accuracy gains. In fact, with our method, we are able to use a codebook of half the size (and half the number of pooled features) while maintaining performance (compare the green and blue curves). It is particularly interesting that, by selecting more features from the over-complete spatial regions, we are able to achieve state-of-the-art performance with a much smaller number of codes (the red curve), which has the potential in time-sensitive or memory-bounded scenarios.

Method	Pooled Features	Accuracy
ours, d=1600	6,400	80.17
ours, d=4000	16,000	82.04
ours, d=6000	24,000	**83.11**
Coates et al. [19], d=1600	6,400	77.9
Coates et al. [19], d=4000	16,000	79.6
Coates et al. [20], d=6000	48,000	81.5
Conv. DBN [64]	N/A	78.9
Improved LCC [121]	N/A	74.5
8-layer Deep NN [17]	N/A	80.49
3-layer Deep NN [21]	N/A	82.0

Table 2.2: Performance on the CIFAR-10 dataset. The first and second blocks compare performance between our method and Coates et al. [19, 20] under similar codebook sizes, where the only difference is the spatial pooling strategy. The third block reports the performance of several state-of-the-art methods in the literature.

2.5.4 Best Performance

Our best performance on the CIFAR-10 dataset was achieved by training a codebook size of 6,000, performing max pooling on over-complete rectangular bins based on a 4×4 grid, and selecting features up to 24,000 dimensions. We also note that the accuracy has not saturated at this number of features, but we would like to test the performance when the number of mid-level features is limited to a reasonable scale. With these settings, we achieved an accuracy of 83.11% on the testing data. To the best of our knowledge, this is the best published result on CIFAR-10 without increasing the training set size by morphing the images.

Table 2.2 lists the performance of several state-of-the-art methods. It is also worth pointing out that, to achieve the same performance, our algorithm usually uses a much lower number of features compared with other well-performing algorithms.

2.5.5 Results on MNIST

We can view the set of learned receptive fields for pooling as a saliency map for classification [52]. To visually show the saliency map and verify its empirical correctness, we applied our method to handwritten digit recognition on the MNIST dataset, on which convolutional deep learning models are particularly effective. To this end, we adopted a similar pipeline as we did for CIFAR-10: dense 6x6 local patches with ZCA whitening are used; a dictionary of size 800 is trained with OMP-1, and thresholding coding with $\alpha = 0.25$ (untuned) is adopted. The features are then max-pooled on over-complete rectangular areas based on a 6×6 regular grid. Note that we used a different coding method from the CIFAR-10 experiment to show that the over-complete spatial pooling method is agnostic of the choice of low-level coding

Method	err%
Baseline [20][a]	1.02
Our Method	**0.64**
Lauer et al. [69]	0.83
Labusch et al. [68]	0.59
Ranzato et al. [91]	0.62
Jarrett et al. [53]	0.53

[a]Our implementation.

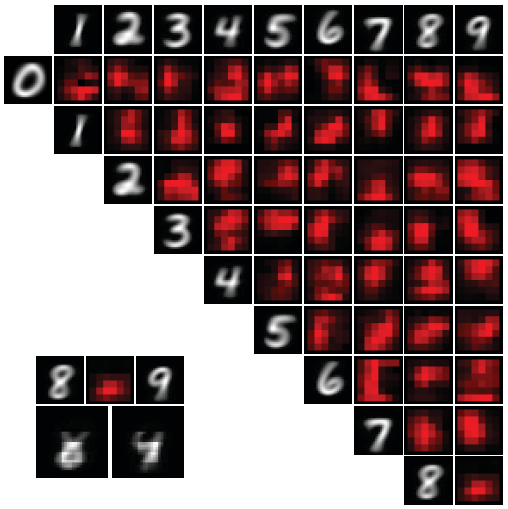


Figure 2.6: Left: Performance comparison (error rate in percentage) on MNIST. Top box: comparison between algorithms using similar pipelines. Bottom box: performance of other related algorithms in the literature. Right: 1-vs-1 saliency maps learned on MNIST. The left-bottom corner plots the mean of digit 8 and 9 multiplied by the corresponding saliency map, showing that the classifier focuses on the bottom part which intuitively also distinguishes the two digits best.

algorithms. Any parameter involved in the pipeline such as SVM regularization weights is tuned on a random 50k/10k split of the training data.

Figure 2.6 shows the 1-vs-1 saliency maps between digits. It can be seen that by learning receptive fields, the classifier focuses on regions where the digits have maximal dissimilarity, e.g., the bottom part for 8 and 9, and the top part for 3 and 5, which matches our intuition about their appearances. For 10-digit classification, we achieved an error rate of 0.64%, on par with several state-of-the-art algorithms (Figure 2.6 left). A gap still exists between our method and the best deep-learning algorithm, and combining receptive learning with deeper structures is future work.

2.5.6 Results on Caltech-101

Lastly, we report the performance of our algorithm compared with SPM on the Caltech-101 dataset in Table 2.3. State-of-the-art performance following similar pipelines are also included in the table. Specifically, we used the same two-step pipeline as proposed by Yang et al. [117]: SIFT features are extracted from 16×16 patches with a stride of 8, and are coded using sparse coding with a codebook of size 1024. For SPM, the coded features are pooled over a pyramid of $1 \times 1, 2 \times 2, 4 \times 4$ regular grids; for a fair comparison we also use the 4×4 regular grid as our base regions, and select the same number of features as SPM uses.

Method	Codebook	Pooling	Performance
ScSPM [117]	1024 (SC)	SPM	73.2±0.54
LCC+SPM [112]	1024	SPM	73.44
Our Method	1024 (SC)	OC	**75.3±0.70**
Boureau et al. [12]	64K	SPM	77.1±0.7
SPM [70]		64.6±0.7	
NBNN [9]		72.8±0.39 (15 training)	
Jarret et al. [53]		65.6±1.0	
RLDA [60]		73.7±0.8	
Adaptive Deconv. Net [122]		71.0±1.0	
Feng et al. [40]		82.6	

Table 2.3: Performance comparison (accuracy in percentage) on Caltech-101. Top: comparison between algorithms using similar pipelines. Bottom: performance of other related algorithms in the literature.

As can be observed in the table, our pooling algorithm outperforms spatial pooling, although a gap still exists between our result and state-of-the-art methods, which uses more complex coding schemes than that we used. The results suggest that coding is a more dominant factor for the performance of Caltech-101. Existing research, especially the Naive Bayes nearest neighbor method [9], has also shown a consistent increase of accuracy with higher-dimensional coding output [12, 118]. However, we still obtain a consistent gain by adopting more flexible receptive fields for pooling, which justifies the effectiveness of the proposed algorithm. Note that the best performance reported by Feng et al. [40] was obtained by jointly learning the pooling operator (p in p-norm pooling) and a per-code spatial saliency map in addition to a larger dictionary, which also follows the idea of learning better spatial information beyond SPM.

2.5.7 Transferring Class-Independent Pooling Knowledge

Beyond classifying existing labels during training, we are also interested in examining whether the learned receptive fields work equally well on unseen classes. While several papers have suggested that simplex cells in V1 performs sparse encoding independent from class labels, and that unsupervised feature learning performs well for the coding step, little is known about the pooling strategy. Learning class-independent pooling knowledge is closely connected to the visual attention model [52], which answers the question "what does an object look like in general".

To examine the performance of our method against new classes, we utilize the CIFAR-100 dataset, which contains 100 categories with 500 training examples per class. We extract features in the same fashion, and train the SVM classifier with learned codes and receptive fields from CIFAR-10. The classification result is compared against the accuracy rate obtained

Classification on	Feature Selection on	Accuracy
CIFAR-100	CIFAR-10	**54.88**
	CIFAR-100	54.83
	Random Selection	54.48±0.25
CIFAR-10	CIFAR-100	78.88
	CIFAR-10	**80.17**
	Random Selection	78.95±0.20

Table 2.4: The performance of transferring pooled feature between CIFAR-10 and CIFAR-100 compared against natively learned features and random selection.

from directly learning the receptive fields on CIFAR-100, and a baseline that does random feature selection from the same set of over-complete features. For a fair comparison, all methods use a codebook size of 1,600 and select 6,400 dimensional features. We also tested learning pooled features on CIFAR-100 and testing on CIFAR-10, and the performances are reported in Table 2.4.

The result we obtained showed a mixed message. While the features are leaned from CIFAR-10, they perform well on the CIFAR-100 dataset, and the performance is even better than natively learned features. For the other direction, transferring learned features does not show a statistically significant difference compared with random selection. A possible explanation of such scenario may be that with less training data per class on CIFAR-100, the feature selection algorithm may suffer more from overfitting than it does on CIFAR-10, reducing the generalization ability of the learned features.

In general, our experiment does show the hope for a better and class-independent pooling strategy. Possible future work may involve utilizing larger-scale image databases, and exploring pooled feature learning in an unsupervised approach, which may further reveal valuable pooling strategies.

2.6 Summary

This chapter focused on analyzing the basic coding and pooling component of the state-of-the-art image feature extraction pipelines. Specifically, we show that smarter algorithms that explicitly take into consideration the pooling stage, whether to find better spatial receptive fields or to find pooling-aware lower level dictionaries, provide significant performance boosts in the final classification accuracies.

Chapter 3

Theoretical Analysis for Feature Learning

One important phenomenon observed in the previous chapter (see *e.g.,* Table 2.2 and Figure 2.5), as well as in related publications [19, 20], is that feature dimension almost always plays a key role in the final classification performance. With higher dimensional features and a simple linear classifier, performance usually appear to be monotonically increasing, although higher dimensionality comes with higher computational costs. It is also noteworthy that with a rather simple coding scheme and dictionary learning, results were in most cases comparable to the widely used but more computationally expensive sparse coding technique [20]. Furthermore, even selecting random dictionaries yielded close to state-of-the-art results. Further work on this domain [30] suggests that the encoding technique used is a proxy to solving sparse coding (but in a simple and faster fashion).

The fact that random dictionaries perform well when operating with large codebook sizes poses interesting questions such as how feature size affects performance. In addition, even though the size of the dictionary (or codebook) is important, the accuracy seems to saturate, which is a phenomenon that was empirically verified in many tasks, and for which we now give a theoretical interpretation by linking random dictionaries with Nyström sampling. In this section, we explain in detail how the feature learning approach could be viewed as a Nyström sampling scheme from a high-dimensional (potentially infinite dimensional) feature space, and then derive proper bounds to model the behavior we observe in the classification experiments. We will use slightly different notations from the previous section, as we will focus on mathematics that is not necessarily tied to specific coding or pooling operations.

3.1 The Nyström Sampling Explanation

Nyström sampling has been proposed as an efficient way to approximate large PSD matrices (such as kernel matrices) by sampling columns of the matrix. Specifically, let $\mathbf{K}$ be an $N \times N$ matrix, the Nyström method defines an approximation as $\mathbf{K}' = \mathbf{E}\mathbf{W}^{+}\mathbf{E}^{\top}$, where $\mathbf{E}$ is a $N \times c$

matrix with the c columns randomly sampled from those of $\mathbf{K}$, and $\mathbf{W}$ is the square $c \times c$ matrix formed by picking the same c columns and rows from $\mathbf{K}$. Such a sampling perspective have been shown to be very effective in kernel machines [123, 23, 67].

We consider forming a dictionary by sampling our training set (although, as discussed below, better techniques exist that lead to further gains in performance). To encode a new data point $\mathbf{x} \in \mathbb{R}^d$, we apply a (generally non-linear) coding function $\mathbf{c}$ so that $\mathbf{c}(\mathbf{x}) \in \mathbb{R}^c$. The standard classification pipeline considers $\mathbf{c}(\mathbf{x})$ as the new feature space, and typically uses a linear classifier on this space. In this section, we consider the threshold encoding function as in [20], $\mathbf{c}(\mathbf{x}) = \max(0, \mathbf{x}^\top \mathbf{D} - \alpha)$, but the derivations are valid for other different coding schemes.

In the ideal case (infinite computation and memory), we encode each sample $\mathbf{x}$ using the whole training set $\mathbf{X} \in \mathbb{R}^{d \times N}$, which can be seen as the best local coding of the training set $\mathbf{X}$, to the extent that overfitting is handled by the classification algorithm. In fact, larger dictionary sizes yield better performance assuming the linear classifier is well regularized, as it can be seen as a way to do manifold learning [112]. We define the new features in this high-dimensional coded space as $\mathbf{C} = \max(0, \mathbf{X}^\top \mathbf{X} - \alpha)$, where the i-th row of $\mathbf{C}$ corresponds to coding the i-th sample $\mathbf{c}(\mathbf{x}_i)$. The linear kernel function between samples i and j is $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{c}(\mathbf{x}_i)^\top \mathbf{c}(\mathbf{x}_j)$. Thus, performing linear classification on the coded features effectively uses the kernel matrix $\mathbf{K} = \mathbf{C}\mathbf{C}^\top$.

In the conventional context of Nyström sampling for kernels, one randomly samples a subset of the columns of $\mathbf{K}$ and then replaces the original matrix $\mathbf{K}$ with a low-rank approximation $\hat{\mathbf{K}}$. However, in our problem, naively applying Nyström sampling to the matrix $\mathbf{K}$ does not save any computation, as every column of $\mathbf{K}$ requires to encode the corresponding feature with the large dictionary of all N samples. However, if we approximate the matrix $\mathbf{C}$ with Nyström sampling to obtain $\mathbf{C}' \approx \mathbf{C}$, we would get an efficient approximation of the kernel matrix as $\mathbf{K}' \approx \mathbf{K}$:

$$\mathbf{C}' = \mathbf{E}\mathbf{W}^{-1}\mathbf{E}^\top, \text{ and} \tag{3.1}$$

$$\mathbf{K}' = \mathbf{C}'\mathbf{C}'^\top = \mathbf{E}\mathbf{W}^{-1}\mathbf{E}^\top\mathbf{E}\mathbf{W}^{-1}\mathbf{E}^\top = \mathbf{E}\mathbf{\Lambda}\mathbf{E}^\top, \tag{3.2}$$

where the first equation comes from applying Nyström sampling to $\mathbf{C}$, $\mathbf{E}$ is a random subsample of the columns of $\mathbf{C}$, and $\mathbf{W}$ the corresponding square matrix with the same random subsample of both columns and rows of $\mathbf{C}$.

We note that in the traditional coding scheme proposed in [20], if the dictionary is taken randomly then $\mathbf{K}_{coding} = \mathbf{E}\mathbf{E}^\top$, and by applying Nyström sampling to $\mathbf{C}$ we obtain almost the same kernel, where the matrix $\mathbf{\Lambda}$ acts as an additional Mahalanobis metric on the coded space. Adding the term $\mathbf{\Lambda}$ seemed to help in some cases, when the dictionary size is small (for example, in the CIFAR10 dataset, classification performance was improved by about 0.5% when $c < 500$.). We refer to the supplementary material to discuss the effect of $\mathbf{\Lambda}$ and how to efficiently find it without explicitly computing the original $N \times N$ matrix.

3.1.1 Error Bounds on the Approximation

Many existing analyses have computed bounds on the error made in estimating $\mathbf{C}$ by $\mathbf{C}'$ by sampling c columns, such as [103, 67], but not between $\mathbf{K} = \mathbf{C}\mathbf{C}^\top$ and $\mathbf{K}' = \mathbf{C}'\mathbf{C}'^\top$, which we aim to analyze in this section. The bound we start with is [67]:

$$||\mathbf{C} - \mathbf{C}'||_F \leq ||\mathbf{C} - \mathbf{C}_k||_F + \epsilon \max(n\mathbf{C}_{ii}), \tag{3.3}$$

valid if $c \geq 64k/\epsilon^4$ (c is the number of columns that we sample from $\mathbf{C}$ to form $\mathbf{E}$, i.e. the codebook size), where k is the sufficient rank to estimate the structure of $\mathbf{C}$, and $\mathbf{C}_k$ is the optimal rank k approximation (given by Singular Value Decomposition (SVD), which we cannot compute in practice).

Fixing k to the value that retains enough energy from $\mathbf{C}$, we get a bound that gives a minimum ϵ to plug in Eqn. 3.3 for every c (sample dictionary size). This gives us a useful bound of the form $\epsilon \geq \hat{M}\left(\frac{1}{c}\right)^{\frac{1}{4}}$ for some constant $\hat{M}$ (that depends on k). Hence:

$$||\mathbf{C} - \mathbf{C}'||_F \leq O + M\left(\frac{1}{c}\right)^{\frac{1}{4}}, \tag{3.4}$$

where O and M constants that are dataset specific.

However, having bounded the error $\mathbf{C}$ is not yet sufficient to establish how the code size will affect the classifier performance. In particular, it is not clear how the error on $\mathbf{C}$ affect the error on the kernel matrix $\mathbf{K}$. Similarly, having a kernel matrix of different quality will affect classification performance. Recent work [23] proves a linear relationship between kernel matrix degradation and classification accuracy. Furthermore, in the supplementary material, we provide a proof that shows the degradation of $\mathbf{K}$ is also proportional to the degradation of $\mathbf{C}$. Hence, the error bound on $\mathbf{K}'$ is of the same form as the one we obtained for $\mathbf{C}$:

$$||\mathbf{K} - \mathbf{K}'||_F \leq O' + M'\left(\frac{1}{c}\right)^{\frac{1}{4}}. \tag{3.5}$$

We briefly prove the bound here. Recall that $\mathbf{K} = \mathbf{C}\mathbf{C}^\top$ and $\mathbf{K}' = \mathbf{C}'\mathbf{C}'^\top$, and since $\mathbf{C}$ and $\mathbf{C}'$ are symmetric, $\mathbf{K} = \mathbf{C}^2$ and $\mathbf{K}' = \mathbf{C}'^2$. Note that the Frobenius norm satisfies subadditivity and submultiplicativity properties [83], i.e.,

$$||A + B||_F \leq ||A||_F + ||B||_F, \text{ and} \tag{3.6}$$

$$||AB||_F \leq ||A||_F||B||_F. \tag{3.7}$$

Thus, we have

$$\begin{aligned} ||\mathbf{K} - \mathbf{K}'|| &= ||\mathbf{C}^2 - \mathbf{C}'^2|| \\ &= ||(\mathbf{C} - \mathbf{C}')\mathbf{C} + \mathbf{C}'(\mathbf{C} - \mathbf{C}')|| \\ &\leq ||(\mathbf{C} - \mathbf{C}')\mathbf{C}|| + ||\mathbf{C}'(\mathbf{C} - \mathbf{C}')|| \\ &\leq ||(\mathbf{C} - \mathbf{C}')||||\mathbf{C}|| + ||\mathbf{C}'||||(\mathbf{C} - \mathbf{C}')|| \\ &\leq ||(\mathbf{C} - \mathbf{C}')||||\mathbf{C}|| + ||\mathbf{C}' - \mathbf{C}||^2 + \\ &\quad + ||\mathbf{C}||||(\mathbf{C} - \mathbf{C}')|| \\ &= ||(\mathbf{C} - \mathbf{C}')||\,(||(\mathbf{C} - \mathbf{C}')|| + 2||\mathbf{C}||) \\ &= \mathcal{O}(||(\mathbf{C} - \mathbf{C}')||) \end{aligned} \tag{3.8}$$

where all the $||.||$ are the Frobenius norms, and where in the last line we assumed that $||(\mathbf{C} - \mathbf{C}')||$ is sufficiently small and $||\mathbf{C}||$ is constant w.r.t. c. Thus, we can expect that the approximation quality of $\mathbf{K}'$ will be similar than $\mathbf{C}'$, and we will further assume that the quality of the kernel approximation $\mathbf{K}'$ will determine the accuracy of the final classifier, which we will also empirically show in the experiments.

We note that the bound above also applies to the case when further steps, such as pooling, is carried out after coding, provided that such steps produce output feature dimensions that have a one-to-one correspondence with the dictionary entries. Pooling over multiple spatial regions does not change the analysis as it could be deemed as concatenating multiple kernel matrices for the data.

3.2 Evaluating Bounds for Learned Features

We empirically evaluate the bound on the kernel matrix, used as a proxy to model classification accuracy, which is the measure of interest. To estimate the constants in the bounds, we do interpolation of the observed accuracy using the first three samples of accuracy versus codebook size, which is of practical interest: one may want to quickly run a new dataset through the pipeline with small dictionary sizes, and then quickly estimate what the accuracy would be when running a full experiment with a much larger dictionary (which would take much longer to run) with our formulation. We always performed Nyström sampling schemes by doing K-means instead of random selection (although the accuracy between both methods does not change too much when c is sufficiently large).

In Figure 3.1 we plot the accuracy (on both train and test sets) on four datasets: CIFAR-10 and STL from vision, and WSJ and TIMIT from speech. For each dataset we used the first three samples to determine the constants given in the bound. One may practically favor this approach to evaluate performance, as small dictionary sizes are fast to try while large dictionary sizes are of interest. The bound is designed to predict training accuracy [23], but we also do regression on testing accuracy for completeness. We note that testing accuracy will in general also be affected by the generalization gap, which is not captured by the bound analysis.

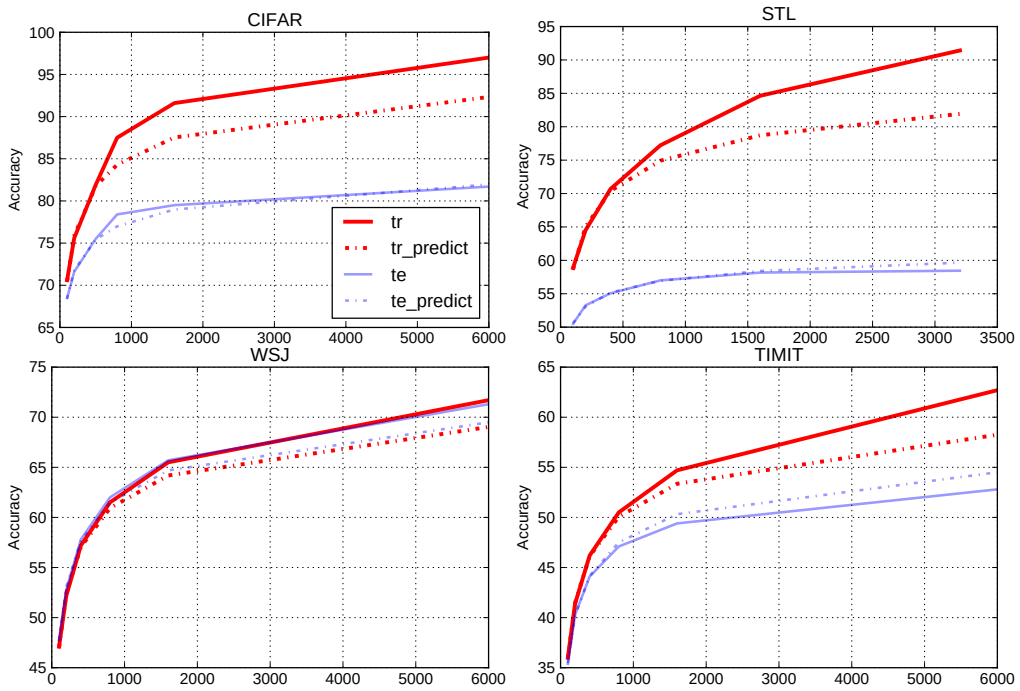


Figure 3.1: The actual training and testing accuracy (solid) and the predicted accuracy using our bound (dashed), on four datasets: CIFAR, STL, WSJ and TIMIT from left to right and top to bottom.

The results show that in all cases, the red dashed line is a lower bound of the training actual accuracy, and follows the shape of the empirical accuracy, predicting its saturation. In the testing case, our model is slightly optimistic when overfitting exists (e.g. STL and TIMIT), but correctly predicts the trend with respect to the number of dictionary entries.

The implication of linking Nyström sampling theory to current learning pipelines has several immediate consequences: first, it clarifies why random sampling or K-means produce very reasonable dictionaries that are able to perform well in terms of classification accuracy [123, 19, 67]; more importantly, due to known bounds such as the one derived in this section, we can model how the codebook size will affect performance by running a few experiments with smaller codebook sizes, and extrapolating to larger (and more computationally expensive to compute) codebook sizes by means of Eq. 3.5, thus predicting accuracies before running potentially long jobs.

We further note that, although our experiment is carried out only by varying the number of codes in the dictionary (to better align speech and vision benchmarks), the pooling operation also falls under the same category: essentially, the whole coding + pooling pipeline could be viewed as a Nyström sampling approach that samples features from the cartesian space of individual codes and individual pooling regions. Also, while we used the term "sampling", the actual feature selection does not necessarily have to be random: research in the

Nyström sampling methods suggests that more data-dependent feature selection approaches, notably K-means, works better than completely randomly sampling features [123, 67], while the theoretical bound still applies to these scenarios. Our work on selecting receptive fields aligns with such work, providing a more informed approach to find better pooled features for classification.

3.3 PADL: Pooling Aware Dictionary Learning

The Nyström sampling view suggests that one could find a better subset of a large (potentially infinite) dictionary to obtain more informative features. In addition, existing work suggests that this could be often done in an efficient way with methods such as clustering. However, current clustering algorithms for dictionary learning [19, 20] only apply to the local coding step, and do not consider the pooling effect. The Nyström sampling insight suggests that simple feature selection approaches may exist that embraces more complex pipelines. As a concise example, we show that by explicitly taking into account the whole pipeline shown in Figure 2.1 to include both local coding and pooling when learning the dictionary, one gets a much more compact feature representation.

Figure 3.2 shows two examples why pooling-aware dictionary learning may be necessary, as local patch-based dictionary learning algorithms often yield similar filters with small translations. Such filters, even when uncorrelated on the patch level, produce highly correlated responses when pooled over a certain spatial region, leading to redundancy in the feature representation.

Observing the effectiveness of clustering methods in patch-based dictionary learning, we propose to learn a final dictionary of size K in two stages: first, we adopt the K-means algorithm to learn a more over-complete starting dictionary of size M ($M >> c$) on patches, effectively "overshooting" the dictionary we aim to obtain. We then perform encoding and pooling using the dictionary, and learn the final smaller dictionary of size c from the statistics of the M-dimensional pooled features.

3.3.1 Post-Pooling Feature Selection

The first step of our algorithm is identical to the patch-based K-means algorithm with a dictionary size M. After this, we can sample a set of image super-patches of the same size as the pooling regions, and obtain the M dimensional pooled features from them. Randomly sampling a large number of pooled features in this way allows us to analyze the pairwise similarities between the codes in the starting dictionary in a post-pooling fashion. We would then like to find a c-dimensional, lower dimensional subspace that best represents the M pooled features.

If we simply would like to find a low-dimensional representation from the M-dimensional pooled features, one would naturally choose SVD to find the K most significant projections of the covariance matrix. With a little abuse of terminology and denoting the matrix of

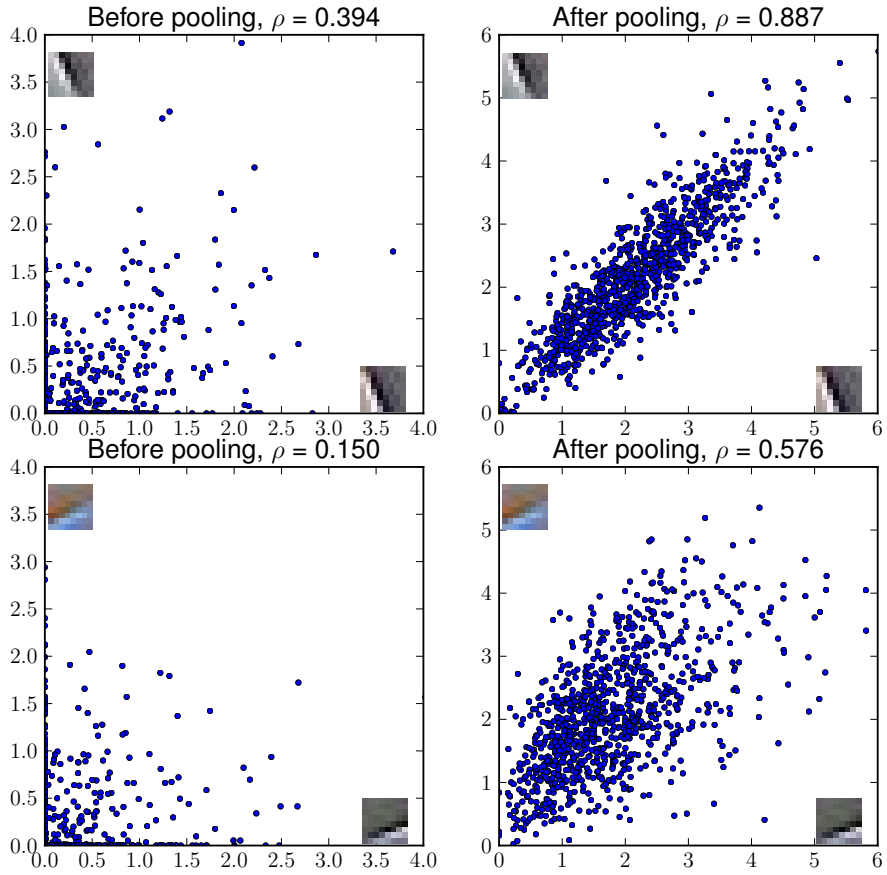


Figure 3.2: Two codes learned from a patch-based K-means algorithm that produce lowly correlated patch-based responses (left), but highly correlated responses after pooling (right). Such phenomenon may root from various causes, such as codes with translational difference (above) and color difference (below).

randomly selected pooled feature as $\mathbf{X}$ where each column is a feature vector, the SVD is carried out as

$$\mathbf{X} \approx \mathbf{U}_c \mathbf{\Lambda}_c \mathbf{V}_c^\top, \tag{3.9}$$

where $\mathbf{R}$ is the covariance matrix computed using the random sample of pooled features, the $M \times c$ matrix U_c contains the left singular vectors, and the $c \times c$ diagonal matrix $\mathbf{\Lambda}_c$ contains the corresponding singular values. The low-dimensional features are then computed as $\mathbf{x}_c = \mathbf{U}_c^\top \mathbf{x}$.

While the "oracle" low-dimensional representation by SVD guarantees the best c-dimensional approximation, it does not meet our goal since the dictionary size is not reduced, as SVD almost always yields non-zero coefficients for all the dimensions. Linearly combining the dictionary entries does not work either due to the nonlinear nature of the encoding algorithm. In our case, we would need the coefficients of only a subset of the features to be

non-zero, so that a minimum number of filters need to be applied during testing time. Various machine learning algorithms aim to solve this, most notably structured sparse PCA [54]. However, these methods often requires a structured sparsity term to be applied during learning, making the training time-consuming and difficult to scale up.

Based on the analysis of the last section, the problem above could again be viewed as a Nyström sampling problem by subsampling the rows of the matrix $\mathbf{X}$ (corresponding to selecting codes from the large dictionary). Empirical results from the Nyström sampling then suggests the use of clustering algorithms to solve this. Thus, we resort to a simpler K-centroids method.

Specifically, we use affinity propagation [42], which is a version of the K-centroids algorithm, to select exemplars from the existing dictionary. Intuitively, codes that produce redundant pooled output (such as translated versions of the same code) would have high similarity between them, and only one exemplar would be chosen by the algorithm. We briefly explain the affinity propagation procedure here: it finds exemplars from a set of candidates where pairwise similarity $s(i,j)$ $(1 \leq i,j \leq M)$ can be computed. It iteratively updates two terms, the "responsibility" $r(i,j)$ and the "availability" $a(i,j)$ via a message passing method following such rules [42]:

$$r(i,k) \leftarrow s(i,k) - \max_{k' \neq k}\{a(i,k') + s(i,k')\} \tag{3.10}$$

$$a(i,k) \leftarrow \min\{0, r(k,k) + \sum\nolimits_{i' \notin \{i,k\}} \max\{0, r(i',k)\}\} \quad (\text{if } i \neq k) \tag{3.11}$$

$$a(k,k) \leftarrow \sum\nolimits_{i' \neq k} \max\{0, r(i',k)\} \tag{3.12}$$

Upon convergence, the centroid that represents any candidate i is given by $\arg\max_k(a(i,k)+r(i,k))$, and the set of centroids $\mathcal{S}$ is obtained by

$$\mathcal{S} = \{k | \exists i, k \text{ s.t. } k = \arg\max_{k'}(a(i,k') + r(i,k'))\} \tag{3.13}$$

And we refer to [42] for details about the nature of such message passing algorithms. The similarity between two pooled dimensions (which correspond to two codes in the starting dictionary) i and code j, as in Eqn. (3.10)-(3.12), is computed as

$$s(i,j) = \frac{2R_{ij}}{\sqrt{R_{ii}R_{jj}}} - 2. \tag{3.14}$$

Note that this is equivalent to the negative Euclidean distance between the coded output i and the coded output j when the outputs are normalized to have zero mean and standard deviation 1. We note that related work such as [18] adopt a similar approach by max-pooling the outputs of similar codes to generate next-layer features in a deep fashion. Our method shares the same merit while focusing on model compression by bounding the computation time in a single layer.

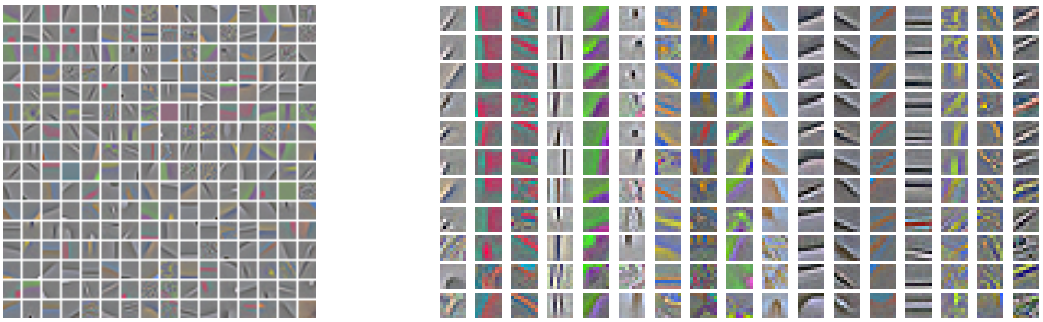


Figure 3.3: Visualization of the learned codes. Left: the selected subset of 256 centroids from an original set of 3200 codes. Right: The similarity between each centroid and the other codes in its cluster. For each column, the first code is the selected centroid, and the remaining codes are in the same cluster represented by it. Notice that while translational invariance is the most dominant factor, our algorithm does find invariances beyond that (e.g., notice the different colors on the last column). Best viewed in color.

Clustering algorithms has shown to be very effective in the context of Nyström sampling [67], and are often highly parallelizeable, easily being scaled up by simply distributing the data over multiple machines. This allows us to maintain the efficiency of dictionary learning. Using a large, overshooting starting dictionary allows us to preserve most information from the patch-level, and the second step prunes away the redundancy due to pooling. Note that the large dictionary is only used during the feature learning time - after this, for each input image, we only need to encode local patches with the selected, relatively smaller dictionary of size c, not any more expensive than existing feature extraction methods.

3.4 Experiments

In this section we empirically evaluate two sets of experiments: using the bound to approximate the classification accuracy, and using the two-staged clustering algorithm to find better pooling invariant dictionaries.

3.4.1 Analysis of Selected Filters

To visually show what codes are selected by affinity propagation, we applied our approach to the CIFAR-10 dataset by first training an over-complete dictionary of 3200 codes following [20], and then performing affinity propagation on the 3200-dimensional pooled features to obtain 256 centroids, which we visualize in Figure 3.3. Translational invariance appears to be the most dominant factor, as many clusters contain translated versions of the same Gabor like code, especially for gray scale codes. On the other hand, clusters capture more than translation: clusters such as column 5 focus on finding the contrasting colors more than

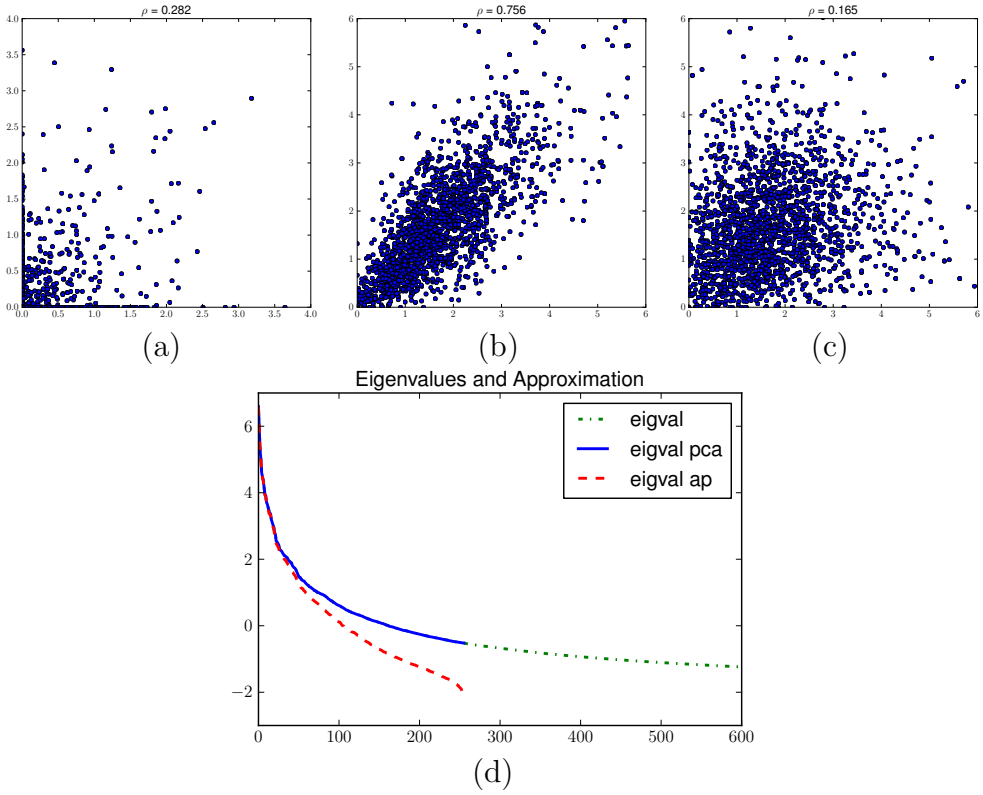


Figure 3.4: (a)-(c): The filter responses before and after pooling: (a) before pooling, between codes in the same cluster (correlation $\rho = 0.282$), (b) after pooling, between codes in the same cluster ($\rho = 0.756$), and (c) after pooling, between the selected centroids ($\rho = 0.165$), (d): the eigenvalues of the approximated matrix (in log scale).

finding edges of exactly the same angle, and clusters such as the last column finds invariant edges of varied color. We note that the selected codes are not necessarily centered, as the centroids are selected solely from the pooled response covariance statistics, which does not explicitly favor centered patches.

We could also verify whether the second clustering stage captures the pooling invariance by checking the statistics of three types of filter responses: (a) pairwise filter responses *before pooling* between codes in the same cluster, (b) pairwise filter responses *after pooling* between codes in the same cluster, and (c) pairwise filter responses after pooling between the selected centroids. The distribution of such responses shown in Figure 3.4 verifies our argument: first, codes that produce uncorrelated responses before pooling may become correlated after the pooling stage (Figure 3.4(a,b)); second, by explicitly considering the pooled feature statistics, we are able to select a subset of the dictionary whose responses are lowly correlated (Figure 3.4(b,c)), preserving more information with a fixed number of codes. In addition, Figure 3.4(d) shows the eigenvalues of the original covariance matrix and those of the approximated matrix, showing that the approximation captures the largest eigenvalues of the original

Task	Learning Method	Accuracy
CIFAR-10 200 codes	K-means	69.02
	2x PADL	70.54 (+1.52)
	4x PADL	71.18 (+2.16)
	8x PADL	71.49 (+2.47)
CIFAR-10 1600 codes	K-means	77.97
	2x PADL	78.71 (+0.74)

Table 3.1: Classification Accuracy on the CIFAR-10 and STL datasets under different budgets.

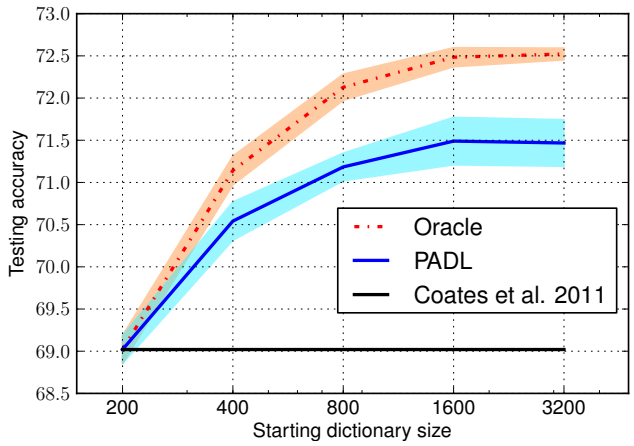


Figure 3.5: Performance improvement on CIFAR when using different starting dictionary sizes and a final dictionary of size 200. Shaded areas denote the standard deviation over different runs. Note that the x-axis is in log scale.

covariance matrix well.

3.4.2 Pooling Invariant Dictionary Learning

To evaluate the improvement introduced by learning a pooling invariant dictionary as in Section 3.3, we show in Figure 3.5 the relative improvement obtained on CIFAR-10 when we use a fixed dictionary size 200, but perform feature selection from a larger overshooting dictionary as indicated by the X axis. The SVD performance is also included in the figure as an "oracle" for the feature selection performance. Learning the dictionary with our feature selection method consistently increases the performance as the size of the original dictionary increases, and is able to get about two thirds the performance gain as obtained by the oracle performance. We note again that SVD still requires the large dictionary to be used and does not save any testing time.

The detailed performance gain of our algorithm on the two datasets, using different

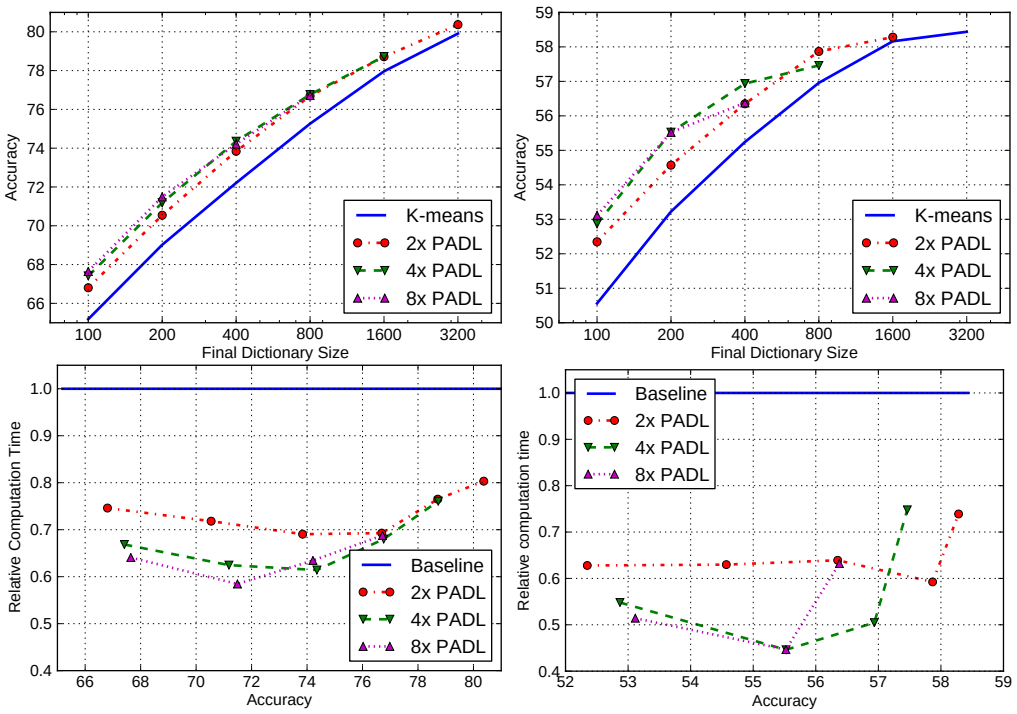


Figure 3.6: Above: accuracy values on the CIFAR-10 (left) and STL (right) datasets under different dictionary size budgets. "nx PADL" means learning the dictionary from a starting dictionary that is n times larger. Below: Relative computation time to achieve the same accuracy using dictionary obtained from PADL.

overshooting and final dictionary sizes, is visualized in Figure 3.6. Table 3.1 summarizes the accuracy values of two particular cases - final dictionary sizes of 200 and 1600 respectively, on CIFAR. Note that our goal is not to get the best overall performance - as performance always goes up when we use more codes. Rather, we focus on two evaluations: (1) how much gain we get given a fixed dictionary size as the budget, and (2) how much computation time we save to achieve the same accuracy.

Overall, considering the pooled feature statistics always help us to find better dictionaries, especially when relatively small dictionaries are used. During testing time, it costs only about 60% computation time with PADL to achieve the same accuracy as K-means does. For the STL dataset, an overly large starting dictionary may lessen the performance gain (Figure 3.6(b)) possibly due to feature selection being more prone to local optimum and the small number of training data (thus more overfitting). However, in general the codebook learned by PADL is consistently better than its patch-based counterpart, suggesting the applicability of the Nyström sampling view in feature learning with a multi-layer structure including spatial pooling.

Finally, we note that due to the heavy-tailed nature of the encoded and pooled features (see the eigendecomposition of Figure 3.4), one can infer that the representations obtained

with a budget would have a correspondingly bounded performance when combined with linear SVMs. We have focused on analyzing unsupervised approaches, but incorporating weakly supervised information to guide feature learning / selection or learning multiple layers of feature extraction would be particularly interesting, and would be a possible future direction.

3.5 Summary

This chapter complements Chapter 2 by revealing the underlying theoretical connection between common dictionary learning algorithms and Nyström sampling. This gives the possibility to transfer knowledge between these two otherwise independent research directions. As a concise example, we proposed an purely unsupervised, pooling-invariant learning algorithm that learns compact codes for better classification.

Chapter 4

Visual Concept Learning

Having analyzed the feature generation pipeline for image classification, in this chapter we move on to a higher-level question: how to learn a visual system that infers latent concepts from exemplar images from that concept, a behavior human are known to perform well? To this end, we introduce the cognitive science aspect of the question, and introduce a novel, practical problem that we call *visual concept learning*. We will then bring together findings in cognitive science and computer vision, using machine vision systems to assign novel images locations within a conceptual hierarchy and a Bayesian generalization model to determine how to generalize from these examples.

The result of such effort is a system that comes closer to human performance than state-of-the-art machine vision techniques. Since no existing dataset adequately tests human-like visual concept learning, we have also collected and made available to the community the first large-scale dataset for evaluating whether machine vision algorithms can learn concepts that agree with human perception and label new unseen images, with ground-truth labeling directly obtained from human annotators. We believe that this new task provides challenges beyond the conventional object classification paradigms.

4.1 The Visual Concept Learning Problem

We will first formally define visual concept learning, with the protocol developed in earlier cognitive science work such as [116]. In our problem, an agent (either a computer system or a human participant) aims to learn a novel visual concept from a few example images presented to the agent. Such images are randomly sampled from this unknown concept, and no negative examples are provided, similar to how human learn novel words from a few examples[1]. The agent then has to indicate whether new "query" image are or are not instances of the target concept.

[1]While one may argue that human receive negative feedbacks as well, such information are often only sought for in an active learning fashion, after the initial concept learning behavior with a few positive examples.

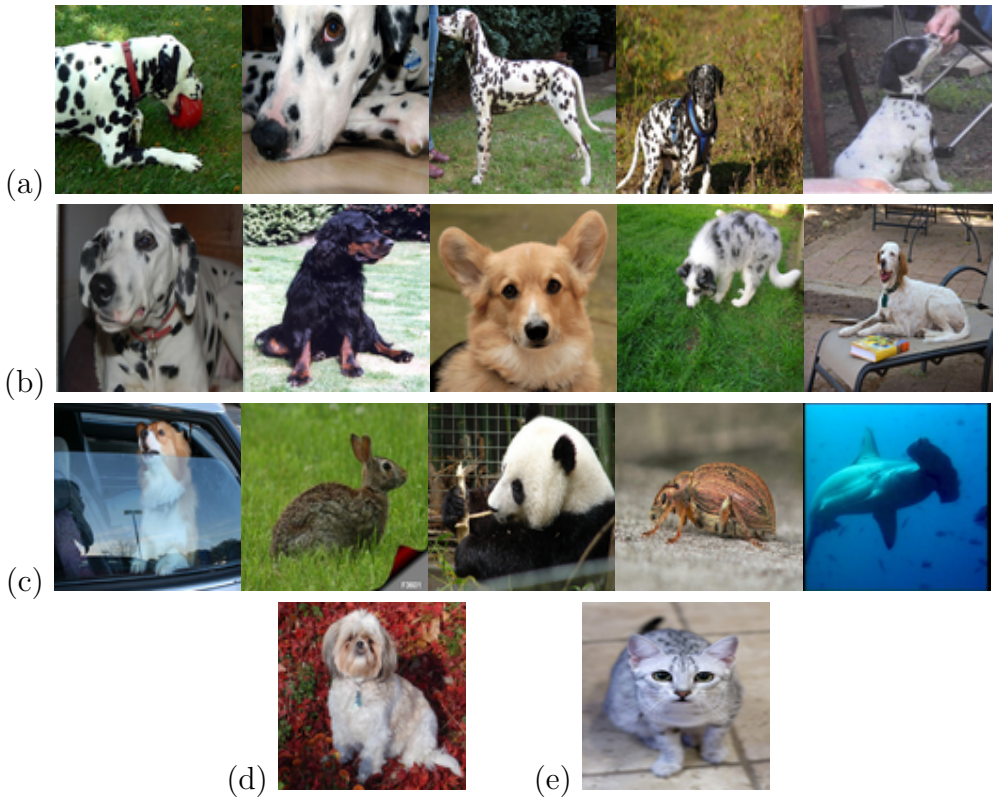


Figure 4.1: Visual concept learning. (a-c): positive examples of three visual concepts. Even without negative data, people are able to learn these concepts: (a) Dalmatians, (b) dogs and (c) animals. Note that although (a) contains valid examples of dogs and both (a) and (b) contain valid examples of animals, people restrict the scope of generalization to more specific concepts, and find it easy to make judgments about whether novel images such as (d) and (e) are instances of the same concepts – the problem we refer to as *visual concept learning*.

A key aspect of this problem is determining the degree to which the concept should be generalized [116] when multiple concepts are logically consistent with the given examples: for example, consider the concepts represented by examples in Figure 4.1 (a-c) respectively, and the problem of predicting whether new images (d-e) belong to them or not. The ground truth from human annotators reveals that the level of generalization varies according to the conceptual diversity, with greater diversity leading to broader generalization. In the examples shown in Figure 4.1, people might identify the concepts as (a) Dalmatians, (b) all dogs, and (c) all animals, but not generalize beyond these levels although no negative images forbids so.

Bayesian models of generalization [1, 104, 116] account for these phenomena, determining the scope of a novel concept (e.g., does the concept refer to Dalmatians, all dogs, or all animals?) in a similar manner to people. However, these models were developed by cognitive scientists interested in analyzing human cognition, and require examples to be manually

labeled as belonging to a particular leaf node in a conceptual hierarchy. This is reasonable if one is asking whether proposed psychological models explain human behavior, but prevents the models from being used to automatically solve visual concept learning problems for a robot or intelligent agent. Machine vision algorithms, on the other hand, still lacks the ability to choose the right level of generalization from the set of valid labels, despite recent successes in large-scale category-level object recognition. We will show in the experiments that state-of-the-art machine vision systems fail to exhibit such patterns of generalization, and have great difficulty learning without negative examples.

4.1.1 Background

Machine vision methods have achieved considerable success in recent years, as evidenced by performance on major challenge problems [29, 33], where strong performance has been obtained for assigning one of a large number of labels to each of a large number of images. However, this research has largely focused on a fairly narrow problem: assigning a label (or sometimes multiple labels) to a single image at a time. This problem is quite different from that faced by a human child trying to learn a new word, where the child is provided with multiple positive examples and has to generalize appropriately. Even young children are able to learn novel visual concepts from very few positive examples [14], something that still poses a challenge for machine vision systems.

Scant attention has been given to the problem of learning a visual concept from a few positive examples as we have defined it. When the problem has been addressed, it has largely been considered from a hierarchical regularization [96] or transfer learning [88] perspective, assuming that a fixed set of labels are given and exploiting transfer or regularization within a hierarchy. Mid-level representations based on attributes [34, 86] focus on extracting common attributes such as "fluffy" and "aquatic" that could be used to semantically describe object categories better than low-level features. Transfer learning approaches have been proposed to jointly learn classifiers with structured regularization [88]. Of all these previous efforts, work that uses object hierarchies to support classification is particularly interesting to our problem scenario. Salakhutdinov et al. [96] proposed learning a set of object classifiers with regularization using hierarchical knowledge, which improves the classification of objects at the leaves of the hierarchy. However, this work did not address the problem of determining the level of abstraction within the hierarchy at which to make generalizations, which is a key aspect of the visual concept learning problem. Deng et al. [28] proposed predicting object labels only to a granularity that the classifier is confident with, but their goal was minimizing structured loss rather than mimicking human generalization.

On the other hand, existing models from cognitive science mainly focus on understanding human generalization judgments within fairly restricted domains. Tenenbaum and colleagues [104, 105] proposed mathematical abstractions for the concept learning problem, building on previous work on models of generalization by Shepard [101]. Xu and Tenenbaum [116] and Abbott et al. [1] conducted experiments with human participants that provided support for this Bayesian generalization framework. Xu and Tenenbaum [116] showed participants

one or more positive examples of a novel word (e.g., "these three objects are Feps"), while manipulating the taxonomic relationship between the examples. For instance, participants could see three toy Dalmatians, three toy dogs, or three toy animals. Participants were then asked to identify the other "Feps" among a variety of both taxonomically related and unrelated objects presented as queries. If the positive examples were three Dalmatians, people might be asked whether other Dalmatians, dogs, and animals are Feps, along with other objects such as vegetables and vehicles. Subsequent work has used the same basic methodology in experiments using a manually collated set of images as stimuli [1]. All of these models assume that objects are already mapped onto locations in a perceptual space or conceptual hierarchy. Thus, they are not able to make predictions about genuinely novel stimuli. Linking such generalization models to direct perceptual input is necessary in order to be able to use this approach to learn visual concepts directly from images.

4.2 Constructing A Large-scale Test Dataset

Existing datasets (PASCAL [33], ILSVRC [5], etc.) test supervised learning performance with relatively large amounts of positive and negative examples available, with ground truth as a set of mutually-exclusive labels. To our knowledge, no existing dataset accurately captures the problem we refer to as visual concept learning: to learn a novel word from a small set of positive examples like humans do. In this section, we describe in detail our effort to make available a dataset for such research.

4.2.1 Test Procedure

In our test procedure, an agent is shown n example images ($n = 5$ in our dataset) sampled from a node (may be leaf nodes or intermediate nodes) from the ImageNet synset tree, and is then asked whether other new images sampled from ImageNet belong to the concept or not. The scores that the agent gives are then compared against human ground truth that we collect, and we use precision-recall curves to evaluate the performance.

We used the ImageNet ILSVRC 2010 synset tree as the beginning point of our data generation procedure for several reasons. First, the ImageNet synset tree is derived from WordNet, which well models the semantics between synsets in a nicely hierarchical structure. Second, ImageNet comes with a large number of image collections manually verified by human whether they belong to the correct synset or not, providing us a large-scale pool for test images. Third, the large number of images allows one to train visual classifiers[2] that identifies images into one of the basic concepts (leaf nodes in the tree), serving as a perceptual basis for concept learning.

From a machine vision perspective, one may ask whether this visual concept learning problem differs from the conventional ImageNet-defined classification problem – identifying

[2]This is analogy to the development of the visual system from a vast number of perceptual input during infant years.

the node from which the examples are drawn, and then answering yes for images in the subtree corresponding to the node, and no for images not from the node. We note that, although we use the nodes in the ImageNet tree to generate examples and queries, the tree itself may not be an accurate hierarchy that matches human perception, and should only be treated as a proxy instead of ground truth. Thus, actual human behavior may differ from what the tree structure implies,In fact, we will show in Section 5.2 that using this approach fails to explain how people learn visual concepts. Human performance in the above task exhibits much more sophisticated concept learning behaviors than simply identifying the node itself, and the latter differs significantly from what we observe from human participants. In addition, with no negative images, a conventional classification model fails to distinguish between nodes that are both valid candidates (*e.g.,* "dogs" and "animals" when shown a bunch of dog images). These make our visual concept learning essentially different and richer than a conventional classification problem.

4.2.2 Automatic Generation of Examples and Queries

Large-scale experimentation requires an efficient scheme to generate test data across varying levels of a concept hierarchy. To this end, we developed a fully-automated procedure for constructing a large-scale dataset suitable for a challenge problem focused on visual concept learning. We used the ImageNet LSVRC [5] 2010 data as the basis for automatically constructing a hierarchically-organized set of concepts at four different levels of abstraction. We had two goals in constructing the dataset: to cover concepts at various levels of abstraction (from subordinate concepts to superordinate concepts, such as from Dalmatian to living things), and to find query images that comprehensively test human generalization behavior. We address these two goals in turn.

To generate concepts at various levels of abstraction, we use all the nodes in the ImageNet hierarchy as concept candidates, starting from the leaf node classes as the most specific level concept. We then generate three more levels of increasingly broad concepts along the path from the leaf to the root for each leaf node in the hierarchy. Examples from such concepts are then shown to human participants to obtain human generalization judgements, which will serve as the ground truth. Specifically, we use the leaf node class itself as the most basic trial type L_0, and select three levels of nested concepts L_1, L_2, L_3 which correspond to three intermediate nodes along the path from the leaf node to the root. We choose the three nodes that maximize the combined information gain across these levels:

$$\mathcal{C}(L_{1\cdots 3}) = \sum_{i=0}^{3} \log(|L_{i+1}| - |L_i|) - \log|L_{i+1}|, \tag{4.1}$$

where $|L_i|$ is the number of leaf nodes under the subtree rooted at L_i, and L_4 is the whole taxonomy tree. As a result, we obtain levels that are "evenly" distributed over the taxonomy tree. Such levels coarsely correspond to the sub-category, basic, super-basic, and super-category levels in the taxonomy: for example, the four levels used in Figure 4.1 are dalmatian, domestic dog, animal, organism for the leaf node dalmatian, and in Figure 4.2(a)

are blueberry, berry, edible fruit, and natural object for the leaf node blueberry. Figure 4.2(b) shows a histogram of the subtree sizes for L_1 to L_3 respectively. For each concept, the five images shown to participants as examples of that concept were randomly sampled from five different leaf node categories[3] from the corresponding subtree in the ILSVRC 2010 test images. Figure 4.1 and 4.2 show such examples. Again, we note that the ImageNet nodes are used as a proxy to generate examples, and may be different from the ground truth for concept learning, which we will collect from human experiments.

To obtain the ground truth (the concepts people perceive when given the set of examples), we then randomly sample twenty query images, and ask human participants whether each of these query images belong to the concept given by the example images. A total of 20 images are randomly sampled as follows: three each from the L_0, L_1, L_2 and L_3 subtrees, and eight images outside L_3. This ensures a complete coverage over in-concept and out-of-concept queries. We explicitly made sure that the leaf node classes of the query images were different from those of the examples if possible, and no duplicates exist among the 20 queries. Note that we always sampled the example and query images from the ILSVRC 2010 test images, allowing us to subsequently train our machine vision models with the training and validation images from the ILSVRC dataset while keeping those in the visual concept learning dataset as novel test images.

4.2.3 Collecting Human Judgements

We created 4,000 identical concepts (four for each leaf node) using the protocol above, and recruited participants online through Amazon Mechanical Turk (AMT, `http://www.mturk.com`) to obtain the human ground truth data. For each concept, an AMT HIT (a single task presented to the human participants) is formed with five example images and twenty query images. The participants were presented with a display where they could easily click what query images belong to the given category or not. Following previous work, participants were told that "Mr. Frog" had picked out some examples of a word in a different language (using a randomly generated word that bears no actual meaning in order to minimize influence from languages), and that "Mr. Frog" needed help picking out the other objects that could be called that word (see Figure 1 for the precise wording). Figure 4.3 shows an example display that a participant could have seen, and possible response from a participant for this trial (all buttons were initialized grey before the participant clicks).

Each HIT was completed by five unique participants, with a compensation of $0.05 USD per HIT. Participants were allowed to complete as many unique trials as they wished. Thus, a total of 20,000 AMT HITs were collected, and a total of 100,000 images were shown to the participants. On average, each participant took approximately one minute to finish each HIT, spending about 3 seconds per query image.

[3]If there are < 5 leaves, each leaf is selected once first, and the remaining counts sampled with replacement.

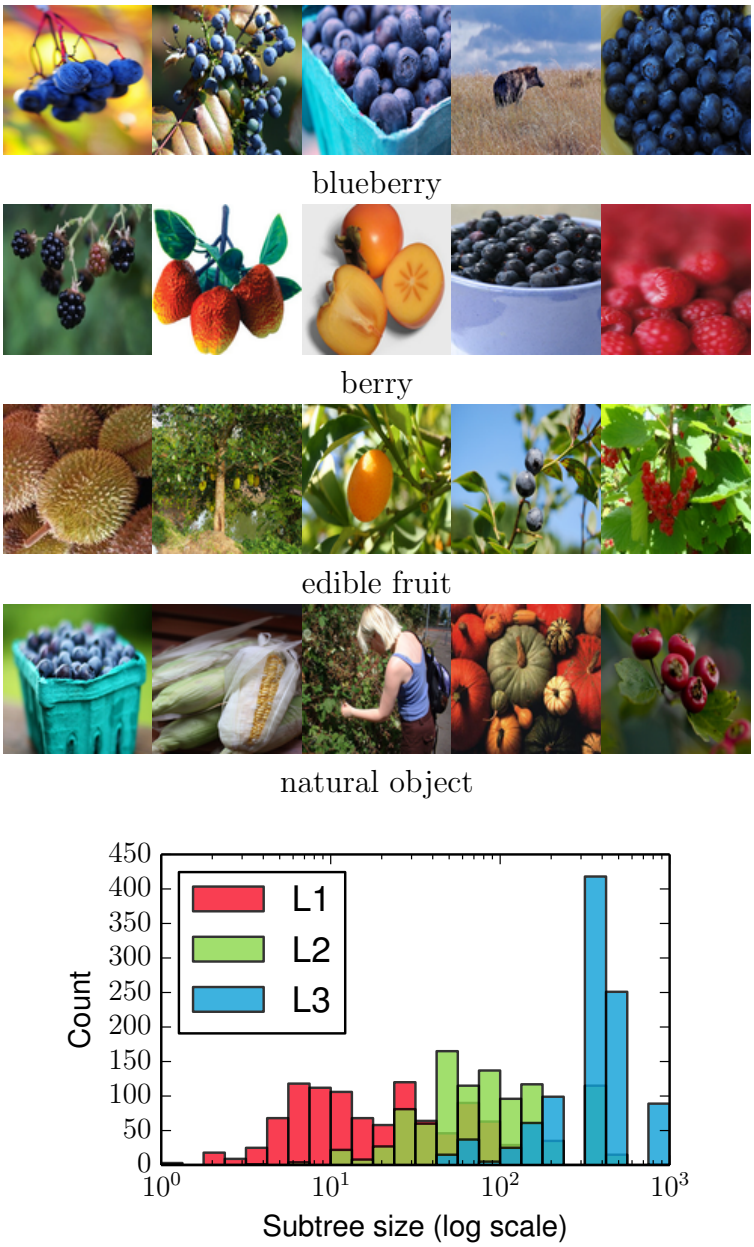


Figure 4.2: Concepts drawn from ImageNet.Top: example images sampled from the four levels for blueberry. Bottom: the histogram for the subtree sizes of different levels of concepts (x axis in log scale).

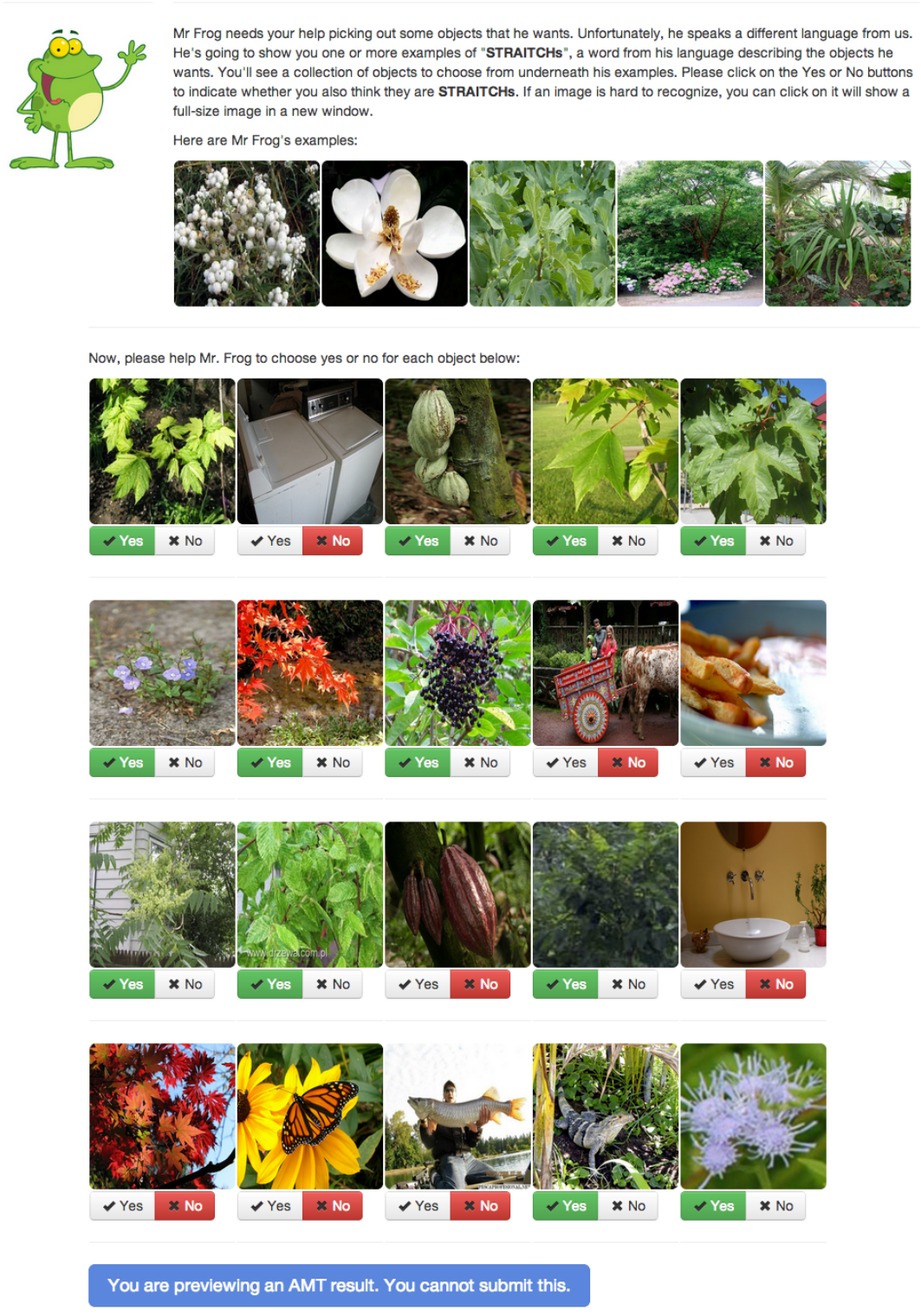


Figure 4.3: An example display that we used for the Mechanical Turk interface, together with the response from the participant.

4.3 Visually-Grounded Bayesian Generalization

In this section, we describe an end-to-end framework which combines Bayesian word learning models and visual classifiers, and is able to perform concept learning with perceptual inputs.

4.3.1 The Bayesian Generalization Model

Prior work on concept learning [116] addressed the problem of generalization from examples using a Bayesian framework: given a set of N examples (images in our case) $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ that are members of an unknown concept $\mathcal{C}$, the probability that a query instance $\mathbf{x}_{\text{query}}$ also belongs to the same concept is given by

$$P_{\text{query}}(\mathbf{x}_{\text{query}} \in \mathcal{C}|\mathcal{X}) = \sum\nolimits_{h \in \mathcal{H}} P_{\text{query}}(\mathbf{x}_{\text{query}}|h) P(h|\mathcal{X}), \tag{4.2}$$

where $\mathcal{H}$ is called the "hypothesis space" – a set of possible hypotheses for what the concept might be. Each hypothesis corresponds to a (often semantically related) subset of all the objects in the world, such as "dogs" or "animals". Given a specific hypothesis h, the probability $P_{\text{query}}(x_{\text{query}}|h)$ that a new instance belongs to it is 1 if x_{query} is in the set, and 0 otherwise, and $P(h|\mathcal{X})$ is the *posterior* probability of a hypothesis h given the examples $\mathcal{X}$. Following previous work, we assume the hypotheses to form a taxonomic hierarchy, where the smallest components (known as *basic concepts*) correspond to the leaf node in the hierarchy. In other words, each node in the ImageNet hierarchy serves as a possible hypothesis. Note that possible concepts are richer than just the collection of hypothesis: for example, one could form a concept like "dogs and cats" by combining the dog and cat subtrees, since Equation (4.2) sums over all hypotheses' posterior probabilities.

Specifically, the posterior distribution over hypotheses is computed using the Bayes' rule: it is proportional to the product of the *likelihood*, $P_{example}(\mathcal{X}|h)$, which is the probability of drawing these examples from the hypothesis h uniformly at random times the *prior* probability $P(h)$ of the hypothesis:

$$P(h|\mathcal{X}) \propto P(h) \prod\nolimits_{i=1}^{N} P_{\text{example}}(\mathbf{x}_i|h). \tag{4.3}$$

To model the conditional probability of an example given a specific hypothesis, we also make the strong sampling assumption that each $\mathbf{x}_i$ is drawn uniformly at random from the set of instances picked out by h. Importantly, this ensures that the model acts in accordance with the "size principle" [104, 105], meaning that the conditional probability of an example given a hypothesis is inversely proportional to the size of the hypothesis, *i.e.,* the number of possible instances that could be drawn from the hypothesis:

$$P_{\text{example}}(\mathbf{x}_i|h) = |h|^{-1} I(\mathbf{x}_i \in h), \tag{4.4}$$

where $|h|$ is the size of the hypothesis and $I(\cdot)$ is an indicator function that has value 1 when the statement is true. We note that the probability of an *example* and that of a *query*

given a hypothesis are different: the former depends on the size of the underlying hypothesis, representing the nature of training with strong sampling. For example, as the number of examples that are all Dalmatians increases, it becomes increasingly likely that the concept is just Dalmatians and not dogs in general even though both are logically possible, because it would have been incredibly unlikely to only sample Dalmatians given that the truth concept was dogs. When asking whether a Dalmatian IS a dog or not, the size principal is then not present: a Dalmatian is no less a dog than a Shih-Tzu, or any other individual dogs.

In addition, the prior distribution $P(h)$ captures biases due to prior knowledge, which favor particular kinds of hypotheses over others (which we will discuss in the next subsection). For example, it is known that people favor basic level object categories such as dogs over subcategories (such as Dalmatians) or super-categories (such as animals). We will describe in detail how the prior distributions and the sizes of hypotheses are formed in our experiments in the next section.

4.3.2 Concept Learning with Perceptual Uncertainty

Existing Bayesian word learning models assume that objects are perfectly recognized, thus representing them as discrete indices into a set of finite tokens. Hypotheses are then subsets of the complete set of tokens and are often hierarchically nested. Although perceptual spaces were adopted in [104], only very simple hypotheses (rectangles over the position of dots) were used. Performing Bayesian inference with a complex perceptual input such as images is thus still a challenge. To this end, we utilize the state-of-the-art image classifiers and classify each image into the set of leaf node classes given in the ImageNet hierarchy, and then build a hypothesis space on top of the classifier outputs. In other words, the classifier outputs could be seen as sufficient statistics of the images.

Such an assumption is of course a simplification of the most general concept learning problem, since it is debatable whether the perception process works by mapping images to a set of discrete leaf node labels and then performing Bayesian generalization on top of it. However, such a process allows us to more directly link the state-of-the-art cognitive science results and computer vision results, verifying the possibility of visual concept learning. In the chapters that follow, we will explore the possibility to perform concept learning directly from a high-dimensional, real-valued perceptual space, in which distances naturally present the semantic relationships between images.

With a little abuse of terminology, in the text that follows, we will denote the image as well as the feature vectors we obtain from them by $\mathbf{x}_i$, and the leaf node label of the image by y_i. Under the assumption above, the conditional probability $P_{\text{example}}(x_i|h)$ then decomposes to the conditional of the leaf node class $P(y_i|h)$ and the conditional of a specific image under that class $P(x_i|y_i)$: from a generative perspective, we will first sample a specific leaf node class from the hypothesis, and then sample an image that belongs to that class as an example. Specifically, we construct the hypothesis space over the image labels using the ImageNet hierarchy, with each subtree rooted at a node serving as a possible hypothesis. The hypothesis sizes are then computed as the number of leaf node classes under the corresponding node,

e.g., the node "animal" would have a larger size than the node "dogs". The large number of images collected by ImageNet allows us to train classifiers from images to the leaf node labels and to estimate the conditional probability $P(x_i|y_i)$, which we will describe shortly in the next section.

4.4 Parameter Estimation

In this section we detail how the various probability scores are defined and estimated, based on both psychological study and machine learning techniques.

4.4.1 Hypothesis Priors

Determining the priors for the various hypotheses is a high-level problem that essentially asks "what object categories do people refer to most often in real life". To this end, we take advantage of the existing research in cognitive science to construct the latent concept space and the prior distribution.

It has been shown that the ImageNet/WordNet hierarchy [38] well models the semantic relations in a psychologically justified tree structure [81], and previous cognitive science has shown promising results in identifying latent concepts (semantically related sets from the universe of objects) for human concept learning [1, 106] at least in a limited scope of testing data. Prior research on psychology and Bayesian generalization [101, 105] have shown that people favor basic-level concepts, which could be well modeled by an Erlang prior with respect to the size $|h|$ of each latent concept, defined as the number of leaf nodes in the subtree rooted at the concept:

$$P(h) = \alpha_h \propto (|h|/\sigma^2)\exp(-|h|/\sigma), \tag{4.5}$$

which favors medium-sized hypotheses corresponding to basic level concepts. Abbotts et al. have shown that a hyperparameter of $\sigma = 200$ matches human behavior best, which we adopt in our work for the prior.

Figure 4.4 shows two such examples along the paths to the ImageNet leaf-node synsets *can opener* and *oriental poppy*. It could be observed that basic level hypotheses, such as "flower" and "tool", have higher probability than overly general hypotheses such as "entity" or overly specific ones such as "oriental poppy", a plausible statistics since we would tend to mention too broad or too specific concepts in the real life.

4.4.2 Image Conditionals

Given a hypothesis, the conditional probability $P(y_i|h)$ follows from assuming strong sampling [105] and the size principle, thus the conditional probability is defined as follows:

$$P(y_i|h) = \beta_{hy_i} = \begin{cases} 1/|h|, & \text{if hypothesis } h \text{ contains leaf node label } y_i \\ 0, & \text{otherwise,} \end{cases} \tag{4.6}$$

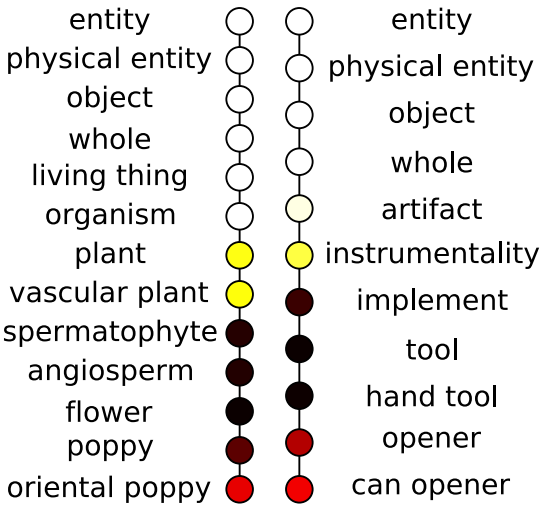


Figure 4.4: The prior probabilities of the hypotheses computed according to existing research, along the path leading to the synsets *oriental poppy* and *can opener* respectively, with darker color indicating higher probability.

where $|h|$ is the size of the hypothesis - the number of leaf node classes under the subtree rooted at the hypothesis[4].

To generate an actual image $\mathbf{x}_i$ from a given class label y_i, it is relatively difficult to fully generative model the conditional probability $P(\mathbf{x}_i|y_i)$ to the pixel level of the images. Thus, we use a mixed generative-discriminative approach by having a classifier trained on all the leaf node objects, and obtain the classifier prediction

$$f(\mathbf{x}_i) = \text{argmax}_j \quad \boldsymbol{\theta}_j^\top \mathbf{x}_i, \tag{4.7}$$

where we assuming that a classifier with parameter $\{\boldsymbol{\theta}_j\}_{j=1}^K$ for K classes is used. The conditional probability is then defined as

$$P(\mathbf{x}_i|y_i) = C_{y_i f(\mathbf{x}_i)}, \tag{4.8}$$

where $\mathbf{C}$ is the confusion matrix of the classifier, and C_{ij} is the probability that an image from object class i is predicted class j by the classifier.

4.5 Terabyte-scale Classifier Training

Recent image classification tasks often involve large amounts of images, making the training of classifiers increasingly difficult. To address this issue, we have developed a distributed,

[4]A keen reader may notice that this further assumes that each leaf node contains the same number of images - which may not be true in the real world. However, the lack of a truly large-scale analysis of object frequencies in the real world renders an accurate estimation of hypothesis sizes unavailable. Thus, we will make a simplified assumption in this thesis.

stochastic optimization toolbox to train large-scale image classifiers. In particular, we used the minibatch approach to perform stochastic gradient descent updates, and utilized the Adagrad [32] algorithm to achieve quasi-Newton performance by accumulating the statistics of the per-iteration gradient estimations, a mechanism shown to work particularly well with vision tasks [27].

Specifically, we focus on training large-scale linear multinomial logistic regressors, which optimizes the following objective function:

$$\mathcal{L}(\boldsymbol{\theta}) = \lambda \|\boldsymbol{\theta}\|_2^2 - \sum\nolimits_{i=1}^{M} \mathbf{t}_i \log \mathbf{u}_i, \tag{4.9}$$

where $\mathbf{t}_i$ is a 0-1 indicator vector where only the y_i-th element is 1, and $\mathbf{u}_i$ is the softmax of the linear outputs

$$u_{ij} = \exp(\boldsymbol{\theta}_j^\top \mathbf{x}_i) / \sum\nolimits_{j'=1}^{K} \exp(\boldsymbol{\theta}_{j'}^\top \mathbf{x}_i), \tag{4.10}$$

where $\mathbf{x}_i$ is the feature for the i-th training image.

To perform training, for each iteration t we randomly sample a minibatch from the data to estimate the gradient $\mathbf{g}_t$, and perform stochastic gradient descent updates. To achieve quasi-Newton performances we adopted the Adagrad [32] algorithm to obtain an approximation of the diagonal of the Hessian as

$$\mathbf{H} = \sigma \mathbf{I} + \sum\nolimits_{n=1}^{t-1} \operatorname{diag}(\mathbf{g}_n \mathbf{g}_n^\top), \tag{4.11}$$

where σ is a small initialization term for numerical stability, and perform parameter upgrade as

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \rho \mathbf{H}^+ \mathbf{g}_t, \tag{4.12}$$

where ρ is a predefined learning rate.

We took advantage of parallel computing by distributing the data over multiple machines and performing gradient computation in parallel, as it only involves summing up the per-datum gradient. As the data is too large to fit into the memory of even a medium-sized cluster, we only keep the minibatch in memory at each iteration, with a background process that pre-fetches the next minibatch from disk during the computation of the current minibatch. This enables us to perform efficient optimization with an arbitrarily large dataset. The overall architecture is visualized in Figure 4.5.

For the image features, we followed the pipeline in [77] to obtain over-complete features for the images. Specifically, we extracted dense local SIFT features, and used Local Coordinate Coding (LCC) to perform encoding with a dictionary of size 16K. The encoded features were then max pooled over 10 spatial bins: the whole image and the 3×3 regular grid. This yielded 160K feature dimensions per image, and a total of about 1.5TB for the training data in double precision format. The overall performance is 41.33% top-1 accuracy and a 61.91% top-5 accuracy on the validation data, and 41.28% and 61.69% respectively on the testing data. For the computation time, training with our toolbox took only about 24 hours with 10 commodity computers connected on a LAN.

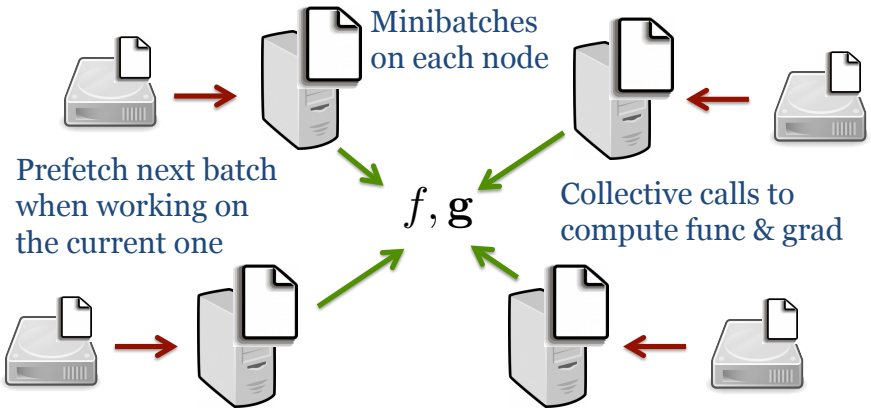


Figure 4.5: The overall architecture of our system.

4.5.1 Confusion Matrix Estimation with One-step Unlearning

Given a classifier, evaluating its behavior (including accuracy and confusion matrix) is often tackled with two approaches: using cross-validation or using a held-out validation dataset. In our case, we note that both methods have significant shortcomings. Cross-validation requires retraining the classifiers multiple rounds, which may lead to high re-training costs. A held-out validation dataset usually estimates the accuracy well, but not for the confusion matrix $\mathbf{C}$ due to insufficient number of validation images. For example, the ILSVRC challenge has only 50K validation images versus 1 million confusion matrix entries, leading to a large number of incorrect zero entries in the estimated confusion matrix (see supplementary material).

Instead of these methods, we propose to approximate its leave-one-out (LOO) error on the training data with a simple gradient descent step to "unlearn" each image to estimate its LOO prediction, similar to the early unlearning ideas [46] proposed for neural networks. We will focus on the use of multinomial logistic regression, which minimizes $\mathcal{L}(\boldsymbol{\theta}) = \lambda\|\boldsymbol{\theta}\|_2^2 - \sum_{i=1}^{M} \mathbf{t}_i \log \mathbf{u}_i$, where $\mathbf{t}_i$ is a 0-1 indicator vector where only the y_i-th element is 1, and $\mathbf{u}_i$ is the softmax of the linear outputs $u_{ij} = \exp(\boldsymbol{\theta}_j^\top \mathbf{x}_i) / \sum_{j'=1}^{K} \exp(\boldsymbol{\theta}_{j'}^\top \mathbf{x}_i)$, with $\mathbf{x}_i$ being the feature for the i-th training image.

Specifically, given the trained classifier parameters $\boldsymbol{\theta}$, it is safe to assume that the gradient $\mathbf{g}(\boldsymbol{\theta}) = \mathbf{0}$. Thus, the gradient for the logistic regression loss when removing a training image $\mathbf{x}_i$ could be computed simply as $\mathbf{g}_{\setminus \mathbf{x}_i}(\boldsymbol{\theta}) = (\mathbf{u}_i - \mathbf{t}_i)\mathbf{x}_i^\top$. Given the Hessian matrix $\mathbf{H}$ at $\boldsymbol{\theta}$, one can perform one-step quasi-Newton least-square update as[5]

$$\boldsymbol{\theta}_{\setminus \mathbf{x}_i} = \boldsymbol{\theta} - \rho' \mathbf{H}^{+} \mathbf{g}_{\setminus \mathbf{x}_i}. \tag{4.13}$$

[5]In practice we used the accumulated matrix $\mathbf{H}$ obtained from Adagrad [32] as a good approximation of the Hessian matrix. See supplementary material for details. We tested the Adagrad $\mathbf{H}$ matrix and the exact Hessian computed at $\boldsymbol{\Theta}$, and found the former to actually perform better, possibly due to its overall robustness.

Note that we put an additional step size ρ' instead of $\rho' = 1$ as would be the case for exact least squares. We set ρ' to the value that yields the same LOO approximation accuracy as the validation accuracy. We use the new parameter $\boldsymbol{\theta}_{\setminus \mathbf{x}_i}$ to perform prediction on $\mathbf{x}_i$ as if $\mathbf{x}_i$ has been left out during training, and accumulate the approximated LOO results to obtain the confusion matrix. We then applied Kneser-Ney [59] smoothing on the confusion matrix for a smoothed estimation.

4.6 Experiments

In this section, we first give a detailed analysis of how well our parameter estimation scheme works, and then describe the experimental protocol adopted to compare our system with human performance, as well as against various baseline algorithms.

4.6.1 Goodness of Estimated Parameters

First, to show that the Adagrad algorithm gives us a reasonable approximation of the Hessian matrix, we computed the ground-truth Hessian matrix at the final parameter, and plot the comparison between the two in Figure 4.6. It could be observed that the Adagrad approximation often gives slightly larger Hessian estimates than the exact Hessian (possibly due to the Large Hessian values before convergence). Empirically, we found that using the Hessian estimation from Adagrad gives better unlearned confusion matrix although the difference is not large (about 0.5 measured by perplexity). We believe that this may be due to the robustness of Adagrad in modeling the general Hessian matrix when the parameter value changes in the parameter space.

As stated in Section 5.3, an good estimation of the confusion matrix $\mathbf{C}$ is crucial for the probabilistic inference. We evaluate the quality of different estimations using the test data: for each testing pair $(y, \hat{y})$, where $\hat{y}$ is the classifier output, its probability is given by the confusion matrix entry $C_{y\hat{y}}$. The perplexity measure [59] then evaluates how "surprising" the confusion matrix sees the testing data results (a smaller value indicates a better fit):

$$perp = \text{Power}\Big(2, \big(\sum\nolimits_{i=1}^{N_{te}} \log_2 C_{y_i \hat{y}_i}\big)/N_{te}\Big),$$

where N_{te} is the number of testing images. Overall, we obtained a perplexity of 46.27 using our unlearning algorithm, while the validation data gave a value of 68.36 and the training data (without unlearning) gave 94.69, both worse than our unlearning algorithm. We refer to the supplementary material for a more complete analysis of the performance of different methods.

Table 4.1 gives the perplexity values of the various sources to obtain the confusion matrix from: the training data (without unlearning), the validation data, and our approach (named as "unlearned"). Two different smoothing approaches are also adopted to test the performance: Laplace smoothing and Kneser-Ney smoothing, with the former smoothes the matrix

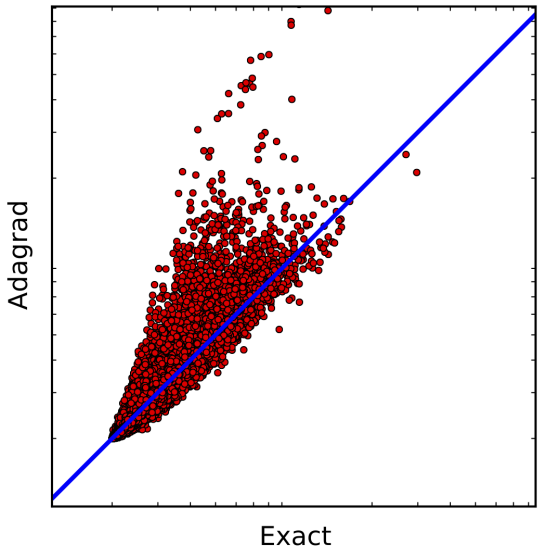


Figure 4.6: The exact hessian and the estimated hessian by Adagrad. Both axes in log scale.

by simply adding a constant term to each entry, and the latter taking a more sophisticated approach and utilizing the bigram information (see [59] for exact math). In general, our approach obtains the best perplexity over all choices.

Figure 4.7 visualizes the confusion matrix entries that are non-zero for the testing data, but missed (*i.e.,* incorrectly predicted as zero) by the methods. Specifically, the dark regions in the figure shows incorrect zero estimates, so the darker the matrix is, the worse the estimation is. We also compute the proportion of zero estimates, defined as the number of non-zero testing entries that are estimated as zero, divided by the total number of non-zero testing entries. The matrix is averaged over 4×4 blocks for better visualization. Overall, matrices estimated from the training and validation data both yield a large proportion (>70%) of incorrect zero entries due to overfitting and lack of validation images respectively, while our method gives a much better estimation with incorrect zero entries <25%. Note that the problem of the remaining sparsity is further alleviated by the smoothing algorithms.

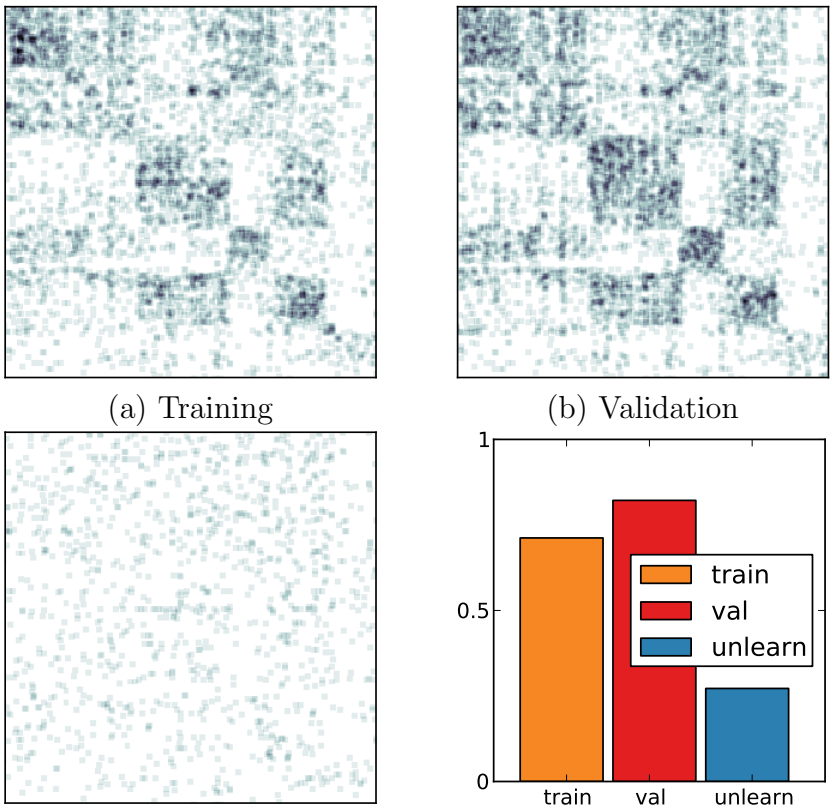
4.6.2 Modeling Human Behavior

To analyze how our visual concept learning model matches human behavior, we use the precision-recall curve, the average precision (AP) and the F_1 score at the point where precision = recall[6] to evaluate the performance and to compare against the human performance,

[6]Empirically (as shown in Figure 4.8), the human participants exhibit approximately the same precision and recall values, so we choose the point on the PR curve where $p = r$, and compute the corresponding F_1 score.

Smoothing	Source	Perplexity
Laplace	training	94.69
	validation	80.52
	unlearned	46.95
Kneser-Ney	training	214.30
	validation	68.36
	unlearned	**46.27**

Table 4.1: The perplexity (lower values preferred) of the confusion matrix estimation methods on the testing data.



(a) Training (b) Validation

(c) Unlearned (d) Proportion of zero estimations

Figure 4.7: Confusion Matrix estimation results. (a)-(c): Visualization of missing estimations (averaged over 4×4 blocks for better readability) for non-zero testing entries obtained from multiple sources. A darker matrix means the estimation misses more entries (a worse estimation) (d): the proportion of missing estimations.

which is calculated by randomly sampling one human participant per distinctive HIT, and comparing his/her prediction against the four others.

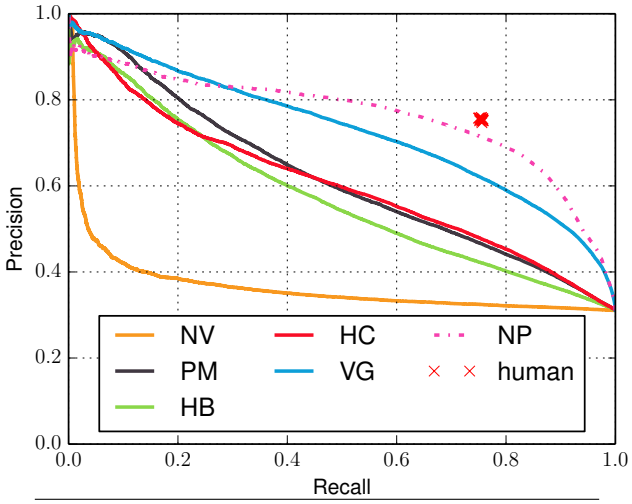
To the best of our knowledge, there are no existing vision models that explicitly handles our concept learning problem. Thus, we compare our vision based Bayes generalization algorithm (denoted by **VG**) described in the previous section against the following baselines, which are reasonable extensions of existing vision or cognitive science models:

1. **Naive vision approach** (NV): this uses a nearest neighbor approach by computing the score of a query as its distance to the closest example image, using GIST features [84].
2. **Prototype model** (PM): an extension of the image classifiers. We use the L_1 normalized classifier output from the multinomial logistic regressors as a vector for the query image, and compute the score as its χ^2 distance to the closest example image.
3. **Histogram of classifier outputs** (HC): similar to the prototype model, but instead of computing the distance between the query and each example, we compute the score as the χ^2 distance to the histogram of classifier outputs, aggregated over the examples.
4. **Hedging the bets extension** (HB): we extend the hedging idea [28] to handle sets of query images. Specifically, we find the subtree in the hierarchy that maximizes the information gain while maintaining an overall accuracy above a threshold ϵ over the set of example images. The score of a query image is then computed as the probability that it belongs to this subtree. The threshold ϵ is tuned on a randomly selected subset of the data.
5. **Non-perceptual word learning** (NP): the classical Bayesian word learning model in [116] assuming a perfect classifier, *i.e.,* by taking the ground-truth leaf labels for the test images. This is not practical in actual applications, but evaluating NP helps understand how a perceptual component contributes to modeling human behavior.

4.6.3 Main Results

Figure 4.8 shows the precision-recall curves for our method and the baseline methods, and summarizes the average precision and F_1 scores. Conventional vision approaches that build upon image classifiers work better than simple image features (such as GIST), which is sensible given that object categories provide relatively more semantics than simple features. However, all the baselines still have performances far from human's, because they miss the key mechanism for inferring the "width" of the latent concept represented by a set of images (instead of a single image as conventional approaches assume). In contrast, adopting the size principle and the Bayesian generalization framework allows us to perform much better, obtaining an increase of about 10% in average precision and F_1 scores, closer to the human performance than other visual baselines.

The non-perceptual (NP) model exhibits better overall average precision than our method, which suggests that image classifiers can still be improved. This is indeed the case, as state-of-the-art recognition algorithms may still significantly underperform human. However, note



Method	AP	F_1 Score
NV	36.37	35.64
PM	61.74	56.07
HC	60.58	56.82
HB	57.50	52.72
NP	76.24	72.70
VG (ours)	72.82	66.97
Human Performance	n.a.	75.47

Figure 4.8: The precision-recall curves of our method and the baseline algorithms. The human results are shown as the red crosses, and the non-perceptual Bayesian word learning model (NB) is shown as magenta dashed lines. The table summarizes the average precision (AP) and F_1 scores of the methods.

that for a system to work in a real-world scenario such as aid-giving robots, it is crucial that the agent be able to take direct perceptual inputs. It is also interesting to note that all visual models yield higher precision values in the low-recall region (top left of Figure 4.8) than the NP model, which does not use perceptual input and has a lower starting precision. This suggests that perceptual signals do play an important role in human generalization behaviors, and should not be left out of the pipeline as previous Bayesian word learning methods do.

4.6.4 Analysis of Per-level Responses

In addition to the quantitative precision-recall curves, we perform a qualitative per-level analysis similar to previous word learning work [1]. To this end, we binarize the predictions at the threshold that yields the same precision and recall, and then plot the per-level responses, *i.e.,* the proportion of query images from level L_i that are predicted positive, given examples from level L_j.

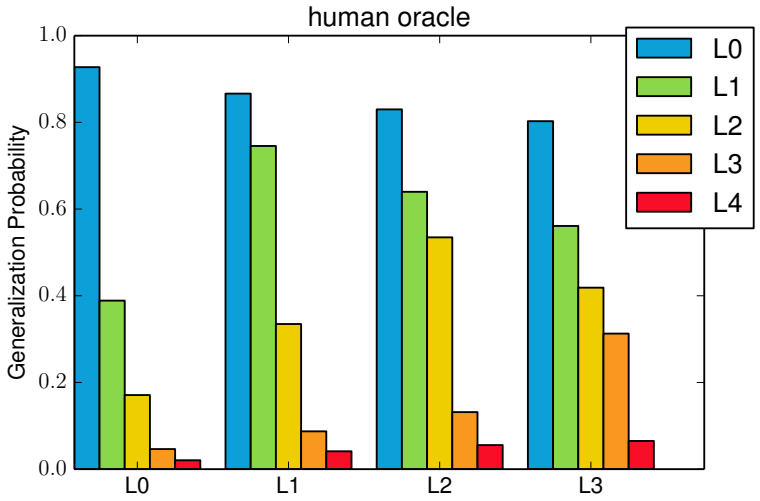
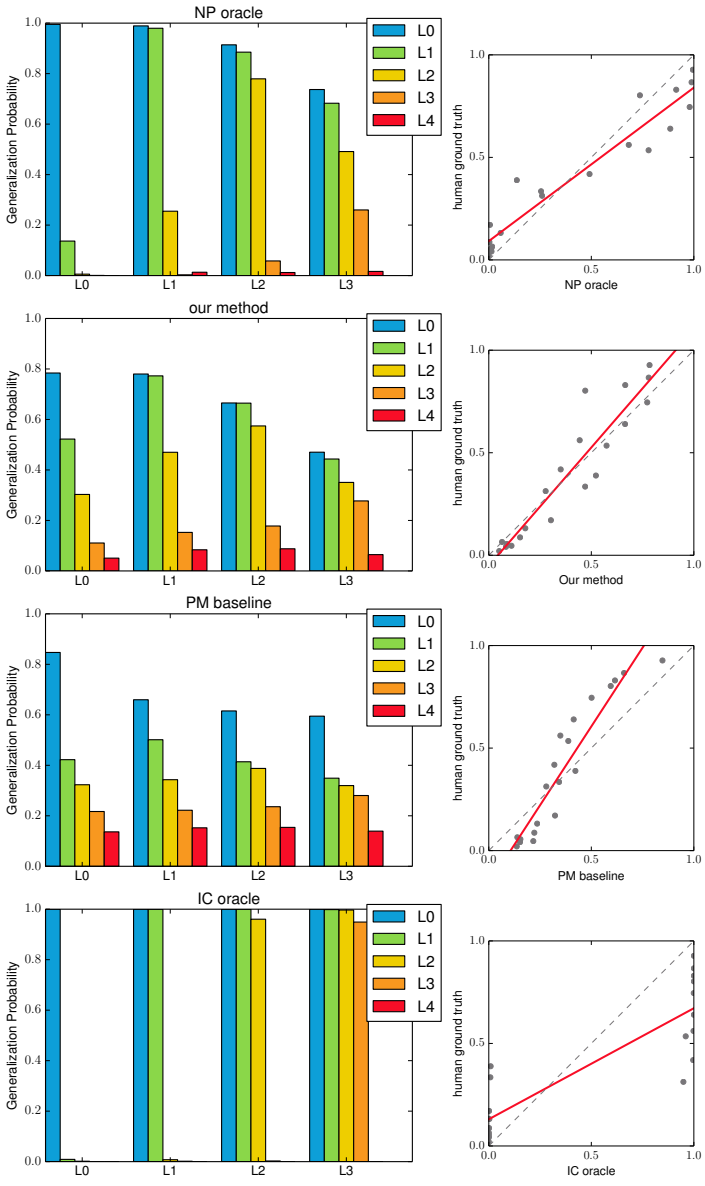


Figure 4.9: Per-level generalization from human participants.

We show in Figures 4.9 and 4.10 the per-level generalization results from human, the NP model, our method, and the PM baseline which best represents state-of-the-art vision baselines. People show a monotonic decrease in generalization as the query level moves conceptually further from the examples. In addition, for queries of the same level, its generalization score peaks when examples from the same level are presented, and drops when lower or higher level examples are presented. The NP model tends to give more extreme predictions (either very low or very high), possibly due to the fact that it assumes perfect recognition, while visual inputs are actually difficult to precisely classify even for a human being. The conventional vision baseline does not utilize the size principle to model human concept learning, and as a result shows very similar behavior with different level of examples. Our method exhibits a good correlation with the human results, although it has a smaller generalization probability for L_0 queries, possibly because current visual models are still not completely accurate in identifying leaf node classes [28].

Last but not least, we examine how well a conventional image classification approach could explain our experimental results. To do so, Figure 4.10(d) plots the results of an image classification (IC) oracle that predicts yes for an image within the ground-truth ImageNet node that the current examples were sampled from and no otherwise. Note that the IC oracle never generalizes beyond the level from which the examples are drawn, and thus, exhibits very different generalization results compared to the human participants in our experiment. Thus, visual concept learning poses more realistic and challenging problems for computer vision studies.



(a) NP Model

(b) Our Method

(c) PM Baseline

(d) IC Oracle

Figure 4.10: Per-level generalization predictions from various methods, where the horizontal axis shows four levels at which examples were provided (L_0 to L_3). At each level, five bars show the proportion of queries form levels L_0 to L_4 that are labeled as instances of the concept by each method. These results are summarized in a scatter plot showing model predictions (horizontal axis) vs. human judgments (vertical axis), with the red line showing a linear regression fit.

4.7 Summary

This chapter proposed a new problem for machine vision – visual concept learning – and presented the first system capable of approaching human performance on this problem. By linking research on object classification in machine vision and Bayesian generalization in cognitive science, we were able to define a system that could infer the appropriate scope of generalization for a novel concept directly from a set of images. This system outperforms baselines that draw on previous approaches in both machine vision and cognitive science, coming closer to human performance than any of these approaches. However, there is still significant room to improve performance on this problem, and we present our visual concept learning dataset as the basis for a new challenge problem for machine vision, going beyond assigning labels to individual objects.

Chapter 5

Latent Task Adaptation with Concept Hierarchies

While the concept learning model proposed above has shown promising performance in modeling human performance, one may ask the question "how this would benefit real-world applications?". This section is devoted to present a real world application - learning multiple specific classification tasks - and demonstrates how a concept learning framework would allow much more flexible classifiers to be deployed.

5.1 Introduction

Recent years have witnessed a growing interest in object classification tasks involving specific sets of object categories, such as fine-grained object classification [35, 62] and home object recognition in visual robotics. Existing methods in the literature generally describe algorithms that are trained and tested on exactly the same task, *i.e.,* we assume the training data and testing data share the same set of object labels. A dog breed classifier is trained and tested on dogs and a cat breed classifier done on cats, without the use of out-of-task images.

However, two observations may render this "one (multi-class) classifier per task" approach suboptimal. First, it's known that using images of related tasks is often beneficial to build a better model for the general visual world [90], which serves as a better regularization for the specific task as well. Second, object categories in the real world are often organized in, or at least well modeled by, a nested taxonomical hierarchy (*e.g.,* Figure 5.1), with classification tasks corresponding to intermediate subtrees in this hierarchy, and recent efforts on the ImageNet challenge [5, 77, 97, 65] have leveraged the use of large-scale data to learn such information. While it is reasonable to train separate classifiers for specific tasks, this quickly becomes infeasible as there are a huge number of possible tasks - any subtree in the hierarchy may be a latent task requiring one to distinguish object categories under the subtree.

Thus, it would be beneficial to have a system which learns a large number of object

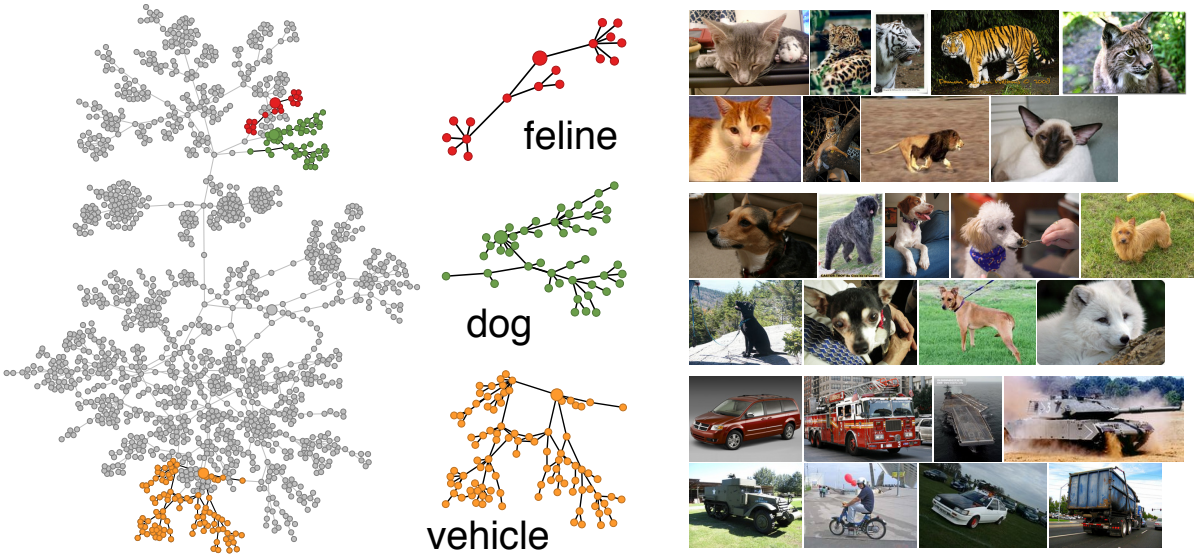
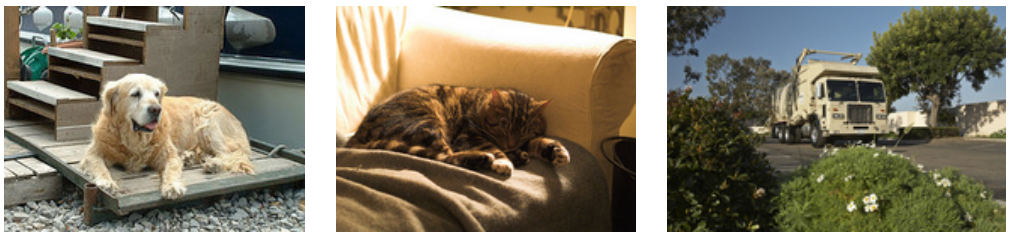


Figure 5.1: Visualization of the ImageNet hierarchy as a tree structure, with three subtrees corresponding to dogs, feline and vehicles highlighted in color. Exemplar images from these three subtrees are presented on the right. Such specific subtrees usually correspond to classification tasks of interest ("which breed of dog is this?").



golden retriever (ice bear)

tabby cat (dungeness crab)

garbage truck (boathouse)

Figure 5.2: Adapting the ImageNet classifier allows us to perform accurate prediction (bold), while the original classifier prediction (in parentheses) suffers from a higher confusion. Note that the classification is carried out together with a set of other images as the task context.

categories in the world, and which is able to quickly adapt to specific incoming classification tasks (subsets of all the object categories) once deployed. We are particularly interested in the scenario where tasks are not explicitly given, but implicitly specified with a set of query images, or a stream of query images in an online fashion. This has practical importance: for example, one may want to have a single mobile app that adapts to plant recognition on a field trip after a few image queries, and that shifts to grocery recognitions when one stops by the grocery store. This is a new challenge beyond simple classification - one needs to discover the latent task using the context given by the queries, a problem that has not been tackled in previous classification problems.

It turns out that this problem is inherently similar to the concept learning problem that we focused on in the previous chapter: while classifying a set of images, one could consider this image set as examples of a latent "task", or "concept", that corresponds to the current application scenario. Thus, in addition to identifying the latent concept itself, which is of interest to visual concept learning, the additional problem is to perform classification under this concept to reveal more fine-grained category labels (such as different species of dogs and birds). This is perfectly applicable under the visual concept learning framework. In this chapter, we will demonstrate one system that achieves this "learn big, predict specific" goal.

Regarding related works along the task adaptation idea, the problem of task adaptation is analogous to, but essentially distinctive from domain adaptation [95, 66]. While domain adaptation aims to model the *perceptual* difference of the training and testing images from the same labels, task adaptation focuses on modeling the *conceptual* difference: different label spaces during training and testing. Additionally, as one is often able to use large amounts of data during training, we assume that the testing tasks involve subsets of labels encountered during training time.

There are several algorithms in image classification that use label hierarchy or structured regularizations to learn better classifiers [96, 47, 43], or to leverage the accuracy and information gain from classifiers [28]. These methods still assume an identical label space for training and testing. The ultimate goal thus remains to be better accuracy on classifying individual images, not to adapt to different tasks during testing time by utilizing contextual information. Better classifiers presented in these papers could, of course, be incorporated in our model to improve the end-to-end performance of task adaptation.

Finally, it is known that contextual information, such as scene context and co-occurring context within a image, could be adopted for better detection [109] or scene understanding [76]. In this work we utilize a novel type of context - *task context* - that is implied by a semantically related group of images.

5.2 Problem Statement

For the sake of clarity, we will first state the task adaptation problem using notations from the previous chapter, and then highlight the connection between the concept learning problem and the latent task adaptation problem.

Formally, we define a classification *task* to be a subset of all the possible object labels that are semantically related (such as all breeds of dogs in ImageNet). During training time, the computer is given all the training images from all these classes, and it will learn one single multi-class classifier. During testing time, a number of query images are randomly sampled from the labels belonging to a task, and the learning algorithm needs to give predictions on these images.

This scenario is much different from the conventional image classification problem setting, as being used in various benchmarks such as Caltech-101 [37] and ILSVRC [5]. Conventionally, we assume a set of mutually exclusive class labels to be presented during both training and testing time. From a probabilistic perspective, it means that the test images are assumed to be drawn i.i.d. from the same label distribution as the training images are. In our problem setting, testing images are mutually related since they together define the task. This makes more practical sense, since one may expect a computer agent to utilize its environment to preform better classification. For example, one could switch to classifying grocery items in a grocery store, and to classifying different animals during a zoo visit. It would be extremely unlikely (and not preferred) for an item in a grocery store to be a giraffe, given the context information.

As stated in the previous section, we are interested in the scenario when the task is *latent*, *i.e.,* only implicitly specified by a set of test images. We introduce two key components for modeling the generative process of test images: a latent task space that defines possible tasks and their probability, and a procedure to sample images given a specific latent task. Specifically, we propose the graphical model in Figure 5.3 which generates a set of N test images when given T possible tasks and K object categories:

1. Sample a latent task h from the task priors $P(h)$ with hyperparameter $\boldsymbol{\alpha}$;
2. For the N images:
   a) Sample an object category y_i from the conditional probability $P(y_i|h;\boldsymbol{\beta}_h)$;
   b) Sample an image from category y_i with $P(x_i|y_i;\boldsymbol{\theta}_{y_i})$.

where the parameters $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\theta}$ are defined as in the previous chapter.

A keen reader may have found the equivalence between a latent *task* described here and a latent *concept* described in the previous chapter. Indeed, they share much in common in the sense that both represent semantically related groups of objects in the real world. Thus, it is natural that one may consider classification in a gradual coarse-to-fine way, with each semantic group forming as a possible task, such as pet breeds of different brands of cars. This is the reason we do not distinguish these two terms, and consider latent concepts as latent tasks as well.

Admittedly, the definition of "task context" is much broader than merely groups of objects derived from a hierarchy. For example, another choice of defining tasks is to relate it to scenes - all objects in an office, or all objects that one may encounter in a street view. In these cases, a task may contain a set of objects that do not belong to, or at least are not

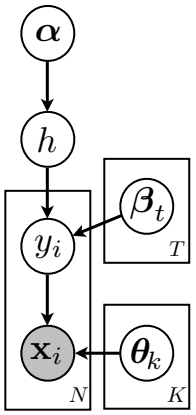


Figure 5.3: The generative model for the latent task and corresponding query images.

well modeled by a taxonomical hierarchy. However, one may reasonably come up with such sets of tasks by looking into meta training data, most notably as co-occurrence of objects in an annotated database such as LabelMe [94]. Thus, we will consider the formation of a latent task space an orthogonal problem, and leave it to future work.

The previous chapter gives the definition of the priors and conditionals, and for the sake of clarity, we will briefly review it here. The prior distribution $P(h)$ is modeled by an Erlang distribution with respect to the size $|h|$ of each latent task, defined as the number of leaf nodes in the subtree rooted at the task:

$$P(h) = \alpha_h \propto (|h|/\sigma^2)\exp(-|h|/\sigma). \tag{5.1}$$

We also choose the hyperparameter σ so that it favors medium-sized tasks, as the last chapter did. The justification is that classification under medium-sized tasks usually correspond to fine-grained recognition problems, which is of particular interest in the literature: an agent that recognizes multiple species of dogs or multiple types of cars may be much more useful than an agent that coarsely classifies dogs vs cars.

Thee conditional probability $P(y_i|h)$ is defined using the strong sampling assumption and the size principle [105]:

$$P(y_i|h) = \beta_{hy_i} = \begin{cases} 1/|h|, & \text{if task } h \text{ contains leaf node label } y_i \\ 0, & \text{otherwise,} \end{cases} \tag{5.2}$$

where $|h|$ is the size of the task. The conditional probability of a specific image given a label is again defined in a mixture of generative and discriminative fashions, using the confusion matrix of the classifier as:

$$P(\mathbf{x}_i|y_i) \propto C_{y_i f(\mathbf{x}_i)}, \tag{5.3}$$

where $\mathbf{C}$ is the confusion matrix of the classifier, C_{ij} is the probability that an image from object class i is predicted class j by the classifier, and $f(\mathbf{x}_i)$ is the classifier prediction

$$f(\mathbf{x}_i) = \underset{j}{\text{argmax}} \quad \boldsymbol{\theta}_j^\top \mathbf{x}_i. \tag{5.4}$$

We note again that the probability $P(\mathbf{x}_i|y_i)$ defined above is a little abuse of terminology, as the partition function is not explicitly given. However, during inference this will only account for a constant bias in the overall likelihood, and will not change the final prediction.

5.3 Linear Time MAP Inference

With the probabilistic model given in Figure 5.3, and given a set of query images to classify as $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$, we formally define the latent task adaptation problem as to jointly identify the hidden task h and the hidden labels $\mathcal{Y} = \{y_1, y_2, \cdots, y_N\}$ that maximizes the posterior probability

$$(\hat{h}, \hat{\mathcal{Y}}) = \operatorname*{argmax}_{h,\mathcal{Y}} P(h, \mathcal{Y}|\mathcal{X}). \tag{5.5}$$

We note that the task adaptation problem focuses on assigning actual labels to both the latent task and the latent labels, while the visual concept learning problem in the last chapter focuses more on generalization, and only provides probability that models the semantic closeness of new query images to the set of example images. This lead to the difference in the inference phase, while the generative model stays the same for both problems.

As most of the parameter estimation are similar to the previous chapter, we will focus on discussing the difference during the inference phase. Specifically, we will propose an efficient inference algorithm that allows one to perform both offline and online adaptation to the task context.

A conventional way to do probabilistic inference with nested latent variables is to use variational inference or Gibbs sampling, both of which find lower bounds of the posterior probability. This, however, may involve multiple iterations over the hidden variables and may be slow. We show that when the latent task space is organized in a directed acyclic graph (DAG) structure (as is the case in the ImageNet data), the exact maximum a posteriori (MAP) estimation (Eqn. (5.5)) could be found with an efficient dynamic programming algorithm that has complexity linear to the number of possible tasks.

We first note that the logarithm of posterior probability in Eqn. 5.5 could be expanded as

$$\log P(h, \mathcal{Y}|\mathcal{X}) \propto \log \alpha_h + \sum\nolimits_{i=1}^{N} \log(\beta_{hy_i} C_{y_i f(\mathbf{x}_i)}). \tag{5.6}$$

Notice that the size constraint defining the latent task space gives us

$$\beta_{hy_i} = \frac{1}{|h|} I(y_i \in h), \tag{5.7}$$

Eqn. 5.5 could further be written as

$$\log \alpha_h - N \log |h| + \sum\nolimits_{i=1}^{n} (\log C_{y_i f(\mathbf{x}_i)} + \log I(y_i \in h)),$$

where one can observe that h and $\mathcal{Y}$ decouples except for the $I(y_i \in h)$ term, which eliminates hypotheses that do not have y_i by setting the log probability to negative infinity. As the

latent tasks are organized as a tree-based hierarchy, we can define auxiliary functions

$$q_i(h) = \max_{\mathcal{Y}} \left[\log C_{y_i f(\mathbf{x}_i)} + \log I(y_i \in h)\right], \tag{5.8}$$

which could be computed recursively as

$$q_i(h) = \max_{\{h' \in child(h)\}} q_i(h'), \tag{5.9}$$

where $child(h)$ is the set of children of h in the tree. Finally, the latent task could be estimated as

$$\hat{h} = \operatorname*{argmax}_h \left[\log(\alpha_h) - N \log |h| + \gamma \sum\nolimits_{i=1}^{N} q_i(h)\right], \tag{5.10}$$

and the corresponding $\hat{y}_i$s could be identified by taking the argmax of the corresponding $q_i(h)$.

We note that we added a hyperparameter γ in the equation above. In practice, simply finding the MAP solution (using $\gamma = 1$) often involves a task that is smaller than the ground truth, as there are two ways to explain the predicted labels: assuming correct prediction and a task of larger size, or assuming wrong prediction and a task of smaller size. The latter is preferred by the size principle, especially for classes with low classification accuracy. We found it beneficial to explicitly add a weight term that favors the classifier outputs using $\gamma > 1$ learned on validation data.

In general, our dynamic programming method runs in $O(TNb)$ time where T is the number of tasks, N is the number of query images, and b is the branching factor of the tree (usually a small constant factor). This complexity is linear to the number of testing images and to the number of latent tasks, and is usually negligible compared to the basic classification algorithm, which runs $O(KND)$ time where K is the number of classes and D is the feature dimension (usually very large).

Finally, one may prefer an online algorithm that could take new images as a stream, performing classification sequentially while discovering the latent task on the fly. We note that our method could be easily adapted to this end. Specifically, $q_i(h)$ serves as the sufficient statistics for the task discovery, and we only need to keep record of the accumulated auxiliary function values seen so far as

$$q_{:n}(h) = \sum\nolimits_{i=1}^{n-1} q_i(h) \tag{5.11}$$

for the n-th image for each task candidate h. This allows us to perform online classification with $O(M)$ memory without storing the full history of images: when a new image $\mathbf{x}_n$ arrives, one simply needs to compute $q_n(h)$ for all hypotheses, and compute its prediction by taking the argmax of $q_n(\hat{h_n})$ using the updated estimate of the task as

$$\hat{h_n} = \operatorname*{argmax}_h \left[\log(\alpha_h) - N \log |h| + \gamma(q_{:n}(h) + q_n(h))\right]. \tag{5.12}$$

Task	Naive	Retraining	FC	Ours
building	55.48	78.67	81.48	**82.19**
dog	35.37	39.94	42.95	**43.76**
feline	47.13	61.07	62.67	**63.54**
home app	50.78	67.30	69.26	**70.52**
vehicle	55.62	61.43	**63.41**	63.28

Table 5.1: Classification accuracy on given tasks (subtrees) of the whole ILSVRC data. See subsection 5.4 for details.

5.4 Analyzing the Necessity of Task Adaptation

An important question to ask is whether we still want to do retraining instead of task adaptation, if one can afford retraining each task. In practice this is certainly impossible with potentially thousands of tasks, but it serves as a proof of concept whether task adaptation benefits overall classification.

To this end, we first analyze the benefits of retraining versus our adaptation method. Specifically, we sampled 5 subtrees from the ILSVRC hierarchy: building, dogs, feline (the superset of cats), home appliance, and vehicle, the subcategories of which are often of interest. Figure 5.2 visualizes the corresponding subtrees for dog, feline and vehicles respectively. Then, we explicitly trained classifiers on these three subtrees only, and compared the retrained accuracy against our adapted classifier with the given task. Since the task is known beforehand, during inference we will simply choose the prediction under the given task. We also tested two baselines: (1) the naive baseline that uses the raw 1000 class predictions, and (2) a forced choice baseline (FC), which simply selects the class under the task that has the largest output from the original classifiers. Table 5.1 summarizes the performance of the algorithms.

It is worth pointing out that retraining the classifiers for the specific tasks does *not* help improve the classification accuracy, although retraining requires additional nontrivial computation cost. In fact, it is always helpful to use out-of-task data to train a larger classifier and then take the subset with forced choice. One possible explanation is that this gives us more information about the general image statistics (similar to a better regularization term), as out-of-task images provide additional negative data during training. Our method further benefits from the statistics from all the classifiers (for in-task and out-of-task classes) in the proposed probabilistic framework. For example, when classifying different dogs, knowing the predicted score of e.g. fox and bears may still benefit the dog classification task under our framework, while the baseline algorithms fail to capture such information. As a result, the proposed algorithm achieves the best adapted accuracy in most cases (only slightly worse than the FC baseline on vehicle).

It is worth noting that such observation will be echoed when we move to the next chapter and analyze the transferability of deep features from state-of-the-art CNN models. This

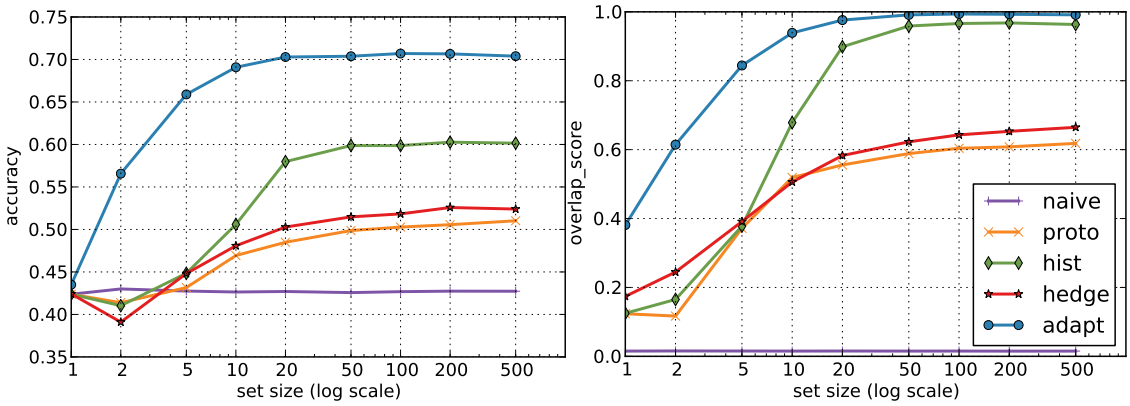


Figure 5.4: Classification accuracy (left) and the task overlap score (right) with different query set sizes for our method and the baselines.

also hints the possibility to learn a general purpose image feature that adapts to multiple application purposes, which is one of the goal (and hope) of deep convolutional models.

5.5 Experiments

We conduct our experiment on the ILSVRC 2010 dataset [5], where both validation and test data are available. To make sure we do not peek into the test images, all hyperparameters and classifiers are learned and validated on the training and validation data.

Also, we note that more comprehensive features and better classification pipelines may lead to better 1-vs-all accuracy on ImageNet, but it is not the main goal of the paper, as we focus on the adaptation on top of the base classifiers. Recent efforts on learning better classifiers, such as the ones presented in [97, 65] could be seamlessly incorporated into our learning framework for general performance increases, and the next chapter will talk about a specific framework that enables on to do so in future research.

5.5.1 Joint Task Discovery and Classification

We analyze the performance when we have the classifier trained on the whole ILSVRC data, and adapt it to an unknown task that is defined by a set of query images. The forced choice option is not available in this case as we do not know the latent task beforehand, and one has to use the semantic relationships between the query images to infer the latent task.

To sample the latent tasks, we used the Erlang prior defined in Section 5.2 from the ImageNet Tree excluding leaf nodes (as leaf nodes would contain only 1 label). We then randomly sampled N query images from the subtree of the sampled task. All query images were randomly selected from the test images of ILSVRC and had not been seen by the classifier training. We varied the value N to assess the quality of task discovery under

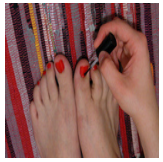
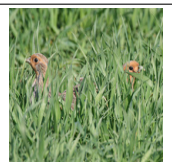
Task: **kitchen app**						entity artifact artifact goods kitchen app
Label	ice maker	espresso maker	primus stove	Dutch oven	ice maker	
Ours	electric range	espresso maker	primus stove	Dutch oven	ice maker	
Baseline	bookcase	web site	carpenter's kit	snail	scanner	
Task: **toiletry**						entity entity entity instrumentality toiletry
Label	lipstick	face powder	nail polish	lotion	hair spray	
Ours	lipstick	face powder	nail polish	lotion	hair spray	
Baseline	toothbrush	dune	bath towel	vending machine	military uniform	
Task: **wood-wind**						entity artifact artifact device reed
Label	bassoon	flute	sax	oboe	sax	
Ours	bassoon	bassoon	sax	oboe	sax	
Baseline	harp	prison	sax	fountain pen	turban	
Task: **game**						entity living thing entity chordate game
Label	ptarmigan	partridge	pheasant	black grouse	quail	
Ours	ptarmigan	partridge	pheasant	black grouse	black grouse	
Baseline	giant panda	orchid	Komodo dragon	Border collie	Newfoundland	

Figure 5.5: Exemplar classification results where incorrect labels are predicted by the base classifiers, but are corrected by our method that benefits from identifying the latent task. Each row shows 5 images from a latent task, and on the right we give the predicted task by different algorithms, ordered and colored as naive, proto, hist, hedge, and adapt. The ground truth label, our prediction and the original classifier's output are provided for each image.

different set sizes. For each query image size N, we created 10,000 independent tasks and reported the average performance here.

To evaluate the goodness of the inferred latent task and the accuracy, we compute the overlap between the ground truth task h and the predicted task $\hat{h}$ as

$$s(h, \hat{h}) = |h \cap \hat{h}|/|h \cup \hat{h}| \times 100\%, \tag{5.13}$$

where $\cap$ and $\cup$ are the intersection and union operations on sets, and $|\cdot|$ denotes the size of a set. For each task, we then compute the accuracy with the predicted labels $\hat{\mathcal{Y}}$ in the standard classification evaluation way. We then report the averaged overlap score and averaged per-task prediction accuracy.

To the best of our knowledge there is no published classification algorithm that is able to identify the latent task, *i.e.,* the intermediate node in the taxonomy hierarchy, given a *set* of query images. Thus, similar to the visual concept learning task in the above chapter, we compare our algorithm against the following baselines that are natural extensions from conventional classification methods:

- **Naive approach**: simply taking the class with the highest prediction score from all the ILSVRC classes.

  Prototype approach: we use the conditional probability $p(y|h)$ as a vector for each task h, and use the task that yields the smallest average distance to each query image (using the classifier outputs) as the predicted latent task. Classification is then performed under this predicted task.

- **Histogram approach**: similar to the prototype approach, but instead of computing pairwise distances to individual query images, we select the task h that yields the smallest χ^2 distance between $p(y|h)$ and the histogram of predictions averaged over all queries.

- **Hedging approach**: we extend the hedging idea [28] to handle sets of query images. Specifically, we find the intermediate node that maximizes the information gain while maintaining an overall accuracy above a threshold ϵ over the set of query images. The corresponding task is then chosen as the predicted latent task. We tune the threshold ϵ on the validation data so that the averaged per-task accuracy is maximized.

We also test an oracle model, in which we explicitly tell the classifier the latent task and perform classification on the subset of labels with the task ground truth. This serves as an upper bound of all methods above, and helps us understand how well different algorithms perform. Regarding the classifier outputs, we used the soft output from the logistic regression for our method, and choose between the soft output and 0-1 hard output for the baseline methods, reporting the better performance of the two here [1].

[1] As a minor note, the hedging method works well with soft outputs, while the prototype and histogram methods prefer soft outputs when the query size is small, and hard outputs when large.

Method	query size=5		query size=100	
	$s(h, \hat{h})$	Accuracy	$s(h, \hat{h})$	Accuracy
Naive	1.54	42.75	1.50	42.68
Proto	8.14	43.16	60.39	50.28
Hist	22.21	44.84	96.61	59.87
Hedging	39.12	44.81	50.34	51.83
Ours	**84.43**	**65.89**	**99.37**	**70.70**
Oracle	100.0	70.36	100.0	70.88

Table 5.2: The average task overlap score and the average accuracy for the algorithms, under query sizes 5 and 100 respectively. All numbers are in percentage. The last row provides the oracle performance in which the ground truth task is given.

As the latent task is inferred with a *set* of images, which directly influences the inference of the latent task, we vary the number of query image sizes and analyze the performance changes. Figure 5.4 shows the performance when we vary the size from 1 to 500, and Table 5.2 summarizes the performance of the methods above with two typical cases: a small query set size (5 images) and a relatively large size (100 image). It could be observed that when we have a reasonable amount of testing queries, identifying the latent task leads to a significant performance gain than the baseline method that does classification against all possible labels, with an increase of near 30% percent. Even with a small query size (such as 5), the performance gain is already noticeably high, indicating the ability of the algorithm to perform task adaptation with very few images from the latent task.

In addition, Figure 5.6 gives more qualitative results showing the benefit of discovering the task context to help classification, where we show multiple tasks under which an incorrectly classified image could be corrected. It could be observed that the base classifier makes some "noble mistakes": a garage that looks like a table and a teapot that looks like a pumpkin. By utilizing the semantic relationship between images presented for the same latent task, the classifier could correct these errors and give more accurate predictions.

5.5.2 Robustness Against Prior Fluctuations

As we use the psychologically derived prior in our model, one possible question is how an inaccurate prior that deviates from human behavior would affect the overall performance. To this end, we change the prior in our model to an uniformed flat prior (*i.e.,* all tasks are equally likely to appear), and examine the change in the classification performance.

Figure 5.7 shows the classification and task prediction performances when the prior mismatch is present. It could be observed that the prior has only minor affect in the performance, mostly when very few query images are used. When the query set size is larger than 10, our algorithm is able to utilize the classifier outputs to correctly overcome the possible bias in

Figure 5.6: More results with our algorithm for different tasks, where the first row are the images, the second row are our predictions, the third row are the predictions made by the base ILSVRC classifier, and the last row is the task name. The task is given implicitly by showing 5 random images (4 not shown here) under the task to the algorithm.

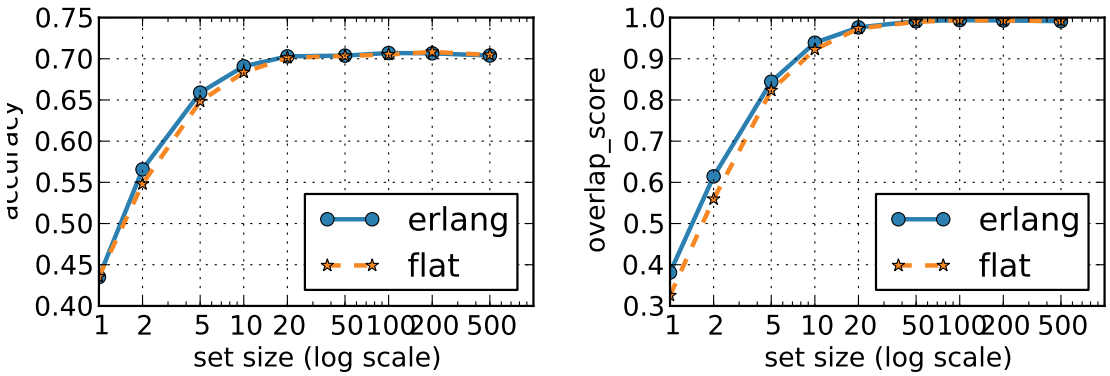


Figure 5.7: Performance comparison between using the ground-truth prior (erlang) and using the uninformed prior (flat). The top figure shows accuracy and the bottom figure shows the overlap score.

the prior probability and achieve almost the same performance as the one using the ground truth task prior.

5.5.3 Online Evaluation

Our final evaluation tests the performance of the proposed method in an online fashion - when images of an unknown task come as a streaming sequence. Intuitively, our algorithm

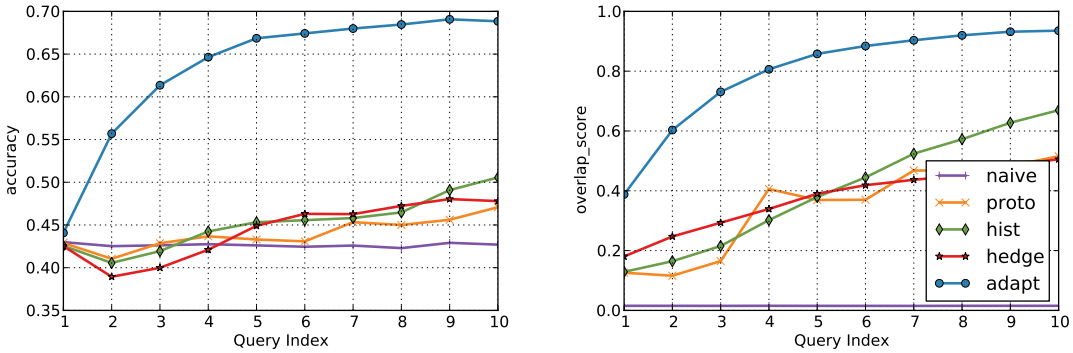


Figure 5.8: Classification accuracy (left) and task overlap score (right) of our online algorithm against baselines. See subsection 5.5.3 for details.

obtains better information about the unknown task as new images arrive, which would in turn increase the classification accuracy. We test such conjecture by evaluating the averaged accuracy of the n-th image, over multiple independent test query sequences that are generated in the same way as described in the previous subsection.

Figure 5.8 shows the average accuracy of the n-th query image, as well as the overlap between the identified task so far and the ground truth task. With the joint probabilistic inference, we obtain a significant performance increase after only a few images. This has particular practical interest, as one may want the computer to quickly adapt to a new task / environment with only a small number of queries. It is worth pointing out that with heuristic task estimation methods (see the baselines in Figure 5.8 left), one may incorrectly assert the latent task, which then hurts classification performance for the first few query images.

5.6 Summary

This chapter provides a concise example on how interdisciplinary research combining vision and cognitive science would allow smarter vision systems to be learned. We focused on a problem - latent task adaptation - that commonly appears in real-world applications but have little existing research on, and showed that an efficient, concept learning inspired framework is able to both discover the latent task context and better predict the categories under the specific task.

Chapter 6

Emergence of Concept-level Information in Deep Networks

With the previous chapters showing the effectiveness of feature learning and concept learning in a more "conventional" pipeline that employs separately designed components, in this chapter we focus on the recently rediscovered deep convolutional neural networks, and show the emergence of object-level representation from a simple, end-to-end trained network.

To this end, I present *Caffe*, an open-source deep learning library developed by me and now maintained by the Berkeley vision group, that allows one to train, test, and deploy state-of-the-art neural networks. Based on this, I will then present empirical validation that a generic visual feature based on a convolutional network weights trained on ImageNet outperforms a host of conventional visual representations on standard benchmark object recognition tasks, including Caltech-101 [36], the Office domain adaptation dataset [95], the Caltech-UCSD Birds fine-grained recognition dataset [113], and the SUN-397 scene recognition database [115].

Further, I will analyze the semantic salience of deep convolutional representations, comparing visual features defined from such networks to conventional representations. Visualization of the semantic clustering properties of deep convolutional features compared to baseline representations reveal that convolutional features appear to cluster semantic topics more readily than conventional features. Also, by tracing back the pixels that contribute to the final predictions, one get an object-centric saliency almost for free, which I will also analyze in this chapter.

As the vision community has only recently rediscovered the power of deep convolutional neural networks, much remains to be analyzed and understood. Thus, compared to previous ones, this chapter is presented in a more exploratory and empirical way, in the hope that it will shed lights guidance for future research on this groundbreaking direction.

6.1 Background

Discovery of effective representations that capture salient semantics for a given task is a key goal of perceptual learning. Performance with conventional visual representations, based on flat feature representations involving quantized gradient filters, has been impressive but has likely plateaued in recent years.

It has long been argued that deep or layered compositional architectures should be able to capture salient aspects of a given domain through discovery of salient clusters, parts, mid-level features, and/or hidden units [48, 41, 125, 102, 65]. Such models have been able to perform better than traditional hand-engineered representations in many domains, especially those where good features have not already been engineered [72]. Recent results have shown that moderately deep unsupervised models outperform the state-of-the art gradient histogram features in part-based detection models [92]. However, unsupervised deep models with more than a few layers have proven difficult to train on large-scale visual category recognition tasks.

Deep models have recently been applied to large-scale visual recognition tasks, trained via back-propagation through layers of convolutional filters [73]. These models perform extremely well in domains with large amounts of training data, and had early success in digit classification tasks [74]. With the advent of large scale sources of category-level training data and efficient implementation with on-line approximate model averaging ("dropout") [65], they have recently outperformed all known methods on a large scale recognition challenge [6].

With limited training data, however, fully-supervised deep architectures with the representational capacity of [65] will generally dramatically overfit the training data. In fact, many conventional visual recognition challenges have tasks with few training examples; e.g., when a user is defining a category "on-the-fly" using specific examples, or for fine-grained recognition challenges [113], attributes [10], and/or domain adaptation [95].

Learning from related tasks also has a long history in machine learning beginning with [15] and [107]. Later works such as [3] developed efficient frameworks for optimizing representations from related tasks, and [2] explored how to transfer parameter manifolds to new tasks. In computer vision, forming a representation based on sets of trained classifiers on related tasks has recently been shown to be effective in a variety of retrieval and classification settings, specifically using classifiers based on visual category detectors [111, 75]. A key question for such learning problems is to find a feature representation that captures the object category related information while discarding noise irrelevant to object category information such as illumination.

Transfer learning across tasks using deep representations has been extensively studied, especially in an unsupervised setting [90, 82]. However, reported successes with such models in convolutional networks have been limited to relatively small datasets such as CIFAR and MNIST, and efforts on larger datasets have had only modest success [71]. We investigate the "supervised pre-training" approach proven successful in computer vision and multimedia settings using a concept-bank paradigm [61, 75, 111] by learning the features on large-scale

data in a supervised setting, then transferring them to different tasks with different labels.

To evaluate the generality of a representation formed from a deep convolutional feature trained on generic recognition tasks, we consider training and testing on datasets known to have a degree of dataset bias with respect to ImageNet, such as SUN-397 scene recognition dataset, and the Office dataset used to evaluate domain adaptation performance [16, 66]. This evaluates whether the learned features could undo the domain bias by capturing the real semantic information instead of overfitting to domain-specific appearances.

6.2 Caffe: A Convolutional Architecture for Fast Feature Embedding

While deep convolutional features have gained much interest in the literature, there exists little, if any, toolboxes that provides a truly off-the-shelf deployment of state-of-the-art models, and it is often nontrivial to replicate experimental results reported in the recent papers without involving substantial graduate student time. To facilitate the wide-spread need for deep convolutional features and analysis, I developed a C++ framework that allows one to efficiently experiment with such architectures. I will elaborate a few key system design choices below, and the reader is encouraged to read more technical details on the Caffe website at `http://caffe.berkeleyvision.org/`.

6.2.1 Key Design Choices

GPUs: Necessary for Training With the large computation needs like the convolutional neural networks have, it is virtually impossible to train networks with CPU only. While systems such as the Google Brain [71] are able to address this by leveraging thousands of machines, the budget for such systems may be overly prohibitive. Thus, I depend highly on GPUs to perform computation, with specific care to hide implementation details from researchers who use them.

I used the Nvidia K20 GPU to perform all experiments mentioned in this work, but note that in terms of vision applications, strict scientific computing features such as ECC do not appear to be significant, and commodity GPUs, such as GTX780, work as well with no noticeable performance difference.

Data Storage With the high throughput of GPUs, it is important that one selects a highly efficient data storage to load data from. For example, laying images on disk and randomly reading them leads to a disk throughput of about 6M/s, while the GPU usually requires about 30M/s data throughput[1]. Thus, I chose to store data in the LevelDB database [2] with

[1] Measured when using the SuperVision model by Krizhevsky et al.

[2] https://code.google.com/p/leveldb/

the Google ProtoBuffer format [3]. When combined together, they provide about 150M/s throughput with commodity hard disks (not solid state drives), allowing one to efficiently utilize the full computation capability of GPUs.

Segregation of Interface and Implementation While GPUs provide an efficient way to carry out computation, managing GPU code is known to be difficult especially for highly customized code. Thus, we designed the system with a clear segregation of the interface from specific implementations. Specifically, we proposed a general wrapper that manages synchronized CPU and GPU memories, and all operations are defined on the wrapper. As a result, one is able to define the network without having to specify which architecture it uses. During runtime, depending on whether the user runs with a GPU or not, the code would seamlessly switch between CPU and GPU code, without the need of user intervention or any reimplementation. The synchronized wrapper also provided possibility to extend to further platforms, such as mobile, as the network is explicitly defined in a device-independent way.

CPUs: Still Hope It is worth noting that while training may be difficult with CPUs, inference is still very affordable with them. In practice, the Caffe CPU implementation is able to run the ImageNet SuperVision model at a speed of 20 milliseconds per frame on a state-of-the-art CPU (with 8 cores) when images are provided in minibatches. This actually allows almost real-time computation, and is valuable since most existing computation architectures may not have GPUs yet.

Implementation of Convolution To perform convolution in an efficient way, we chose a path that is memory heavy: each local patch is expanded to a separate vector, and the whole image is converted to a matrix whose rows correspond to the multiple locations where filters will be applied. This effectively converts the convolution to a matrix-matrix multiplication (gemm) call, making it possible to utilize the commonly available, highly optimized BLAS (CUBLAS on Nvidia GPUs) libraries for dense computation.

While one may question the speed of such operations, we note that the most time-consuming part of convolution is the dense gemm. Assuming we have an image of size $N \times N$, a filter size of $K \times K$ and a total number of F filters, then the matrix expansion mentioned above will only account for $O(N^2K^2)$ complexity while the matrix multiplication complexity is $O(N^2K^2F)$. In the case of deep networks, F is usually very large (e.g. 256), effectively amortizing the expansion overhead. In practice, we observed a speedup of about 1.3 times on K20 compared to cuda-convnet[4], which is optimized for GTX580.

Notes on Multi-GPU computation The current Caffe implementation does not utilize multiple GPUs, but as a general note, it is possible that multiple GPUs on the same machine could be used for faster computation - effectively, one may distribute the minibatches during

[3] https://code.google.com/p/protobuf/

[4] `https://code.google.com/p/cuda-convnet/`

training to different GPUs in a synchronous fashion, with MPI-like reduce and broadcast calls after each minibatch. This would be possible with a high-speed inter-GPU memory access, which is made possible in the most recent GPUs. Combining different GPUs on different machines may be more complex, as there are multiple latency issues (such as the network speed) that needs to be accounted for. Related work such as [22] have shown promising results with custom, fast networks, and may be a valuable engineering direction to pursuit in the future.

6.2.2 Training Details

As the underlying architecture for higher-level analysis, I adopted the deep convolutional neural network architecture proposed by Krizhevsky *et al.* [65], which won the ImageNet Large Scale Visual Recognition Challenge 2012 with a top-1 validation error rate of 40.7%[5]. This model is chosen due to its performance on a difficult 1000-way classification task, hypothesizing that the activations of the neurons in its late hidden layers may serve as very strong features for a variety of object recognition tasks. Its inputs are the mean-centered raw RGB pixel intensity values. These values are forward propagated through five convolutional layers (with pooling and ReLU non-linearities applied after certain convolutional layers) and three fully-connected layers to determine its final neuron activities: a distribution over the task's 1000 object categories.

I refer to the original paper for a detailed discussion of the architecture and training protocol, which we closely followed with the exception of a few small differences in the input data: (1) I ignored the images' original aspect ratio and warp them to 256×256, rather than resizing and cropping to preserve the proportions; (2) Images are cropped to 227×227 rather than 224×224 as in the original paper, due to the difference in the convolutional layers' implementations; (3) I did not perform data augmentation by adding random multiples of the principle components of the RGB pixel values throughout the dataset, which was used as a way of capturing invariance to changes in illumination and color. For training, 90 epochs through the data are carried out, with an initial learning rate of 0.01 and dropped by a factor of 10 after epochs 25, 50, and 75.

The model, when trained with Caffe, took about 6 days to converge on a single K20 GPU. The final model achieved a top-1 error of 41.7%, which is 1% shy of the performance reported in [65]. According to the authors, the data augmentation accounts for about 1% performance difference, which explains most of the networks' performance discrepancy.

[5]The model entered into the competition actually achieved a top-1 validation error rate of 36.7% by averaging the predictions of 7 structurally identical models that were initialized and trained independently. We trained only a single instance of the model; hence we refer to the single model error rate of 40.7%.

Architecture	Forward Pass	Backward Pass
GPU	2.38	4.42
CPU	28.76	54.99

Table 6.1: The computation time of both the forward and backward pass, on both GPUs and CPUs, using a batch size of 256. The computation time reported is on each image in milliseconds, averaged over 10 independent batches.

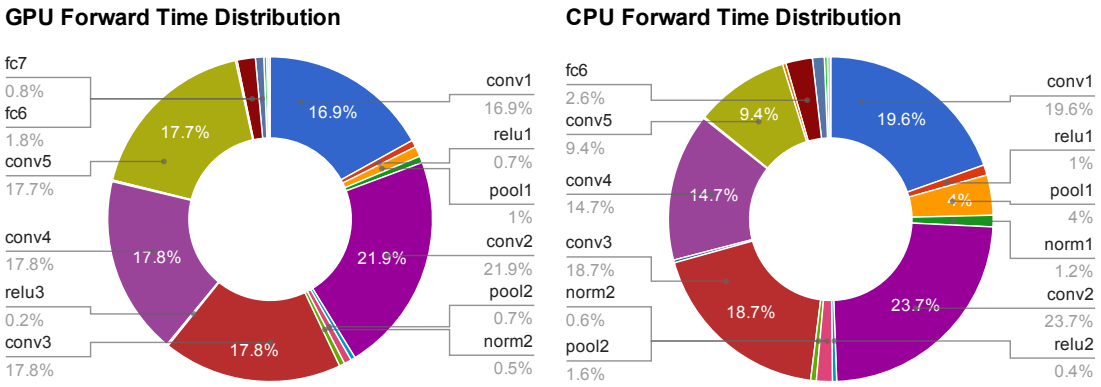


Figure 6.1: Computation time distribution of individual layers, on both GPUs and CPUs for the forward pass.

6.3 Time Analysis

It is of particular interest to analyze the time distribution over different layers, and over different computation architectures. In our earlier publication [31], we presented a preliminary analysis of per-layer speed distributions[6]. In this section, I will provide a more complete analysis, which hopefully provides more insights into further optimization of the learned networks.

I first compare the computation time of both the forward and backward pass, on both GPUs and CPUs, using a batch size of 256 which is typical during training time. Table 6.1 summarizes the computation time of each component. In general, using a GPU speeds computation by about 10 times, both in the forward and the backward pass. Considering the fact that training the ImageNet takes about a week as also independently reported by multiple research papers, GPUs do serve as a more cost-effective solution, especially considering the fact that computation could be further speeded up with multiple GPUs on the same computer.

Further, Figure 6.1 shows the computation time distribution of individual layers on both GPUs and CPUs for the forward pass. The backward pass distribution is similar so they

[6]In all experiments below, I will omit the data layer, whose speed is largely dependent on hardware such as hard disk read speed, and is of less interest.

are omitted here. It could be observed that both platforms spend most computation time on dense convolution and fully connected layers, but the GPU is better capable of handling layers that are memory-bounded, such as pooling and normalization, possibly due to the high memory bandwidth inside the GPU.

It is usually the case that computation is carried out in minibatches. This comes due to 2 reasons: (a) the stochastic gradient descent itself has been shown to work well with minibatches, and (b) minibatches usually amortizes certain computation such as converting matrix-vector multiplication to matrix-matrix multiplication, making computation more efficient than single inputs.

To analyze how the batch size affects the overall computation time, I will use the forward pass only with the GPU computation, and vary the batch size from 1 to 256, which is the range one usually chooses batch sizes from. The absolute time spent by each layer per image is shown in Figure 6.2. Note that we only visualize the eight most time-consuming layers, and omit computationally inexpensive layers such as pooling and ReLU. In total, the rest of the layers takes less than 5% of the total computation time (see Figure 6.2)

Similar to what have been shown in [31], when single images are fed into the network, the fully-connected layers takes most computation time, almost three times as expensive as the convolution layers. However, with minibatches, the computation time of the fully connected layers are quickly amortized, eventually only accounting for less than 5% of the total computation time.

Based on the results in the two subsections above, I make the following notes:

1. In a deployed system such as an online API or a video processing application, one may group requests in batches (such as aggregating multiple frames) to more efficiently use the computation resource.

2. When single-image processing is a must, optimization on the fully connected layers are necessary, especially to reduce the number of hidden units. It is worth pointing out that the Nyström sampling perspective I discussed in earlier chapters may be particularly helpful in this case.

3. Convolutional layers, even when carried out with single images, is already efficient enough. At some level, this contradicts with common belief that minibatches have to be used for efficient computation. As a result, it is possible to use convolutional layers on a very large image with little loss of efficiency.

4. While most existing detection approaches attack the detection problem in a separately designed paradigm, which first finds bounding box proposals with low-level cues and then perform classification on each bounding box, the efficiency of convolutional layers makes it applicable to perform detection in a sliding window fashion. Note that in detection, the fully-connected layers could be trivially converted to convolutional layers, making the whole pipeline highly efficient.

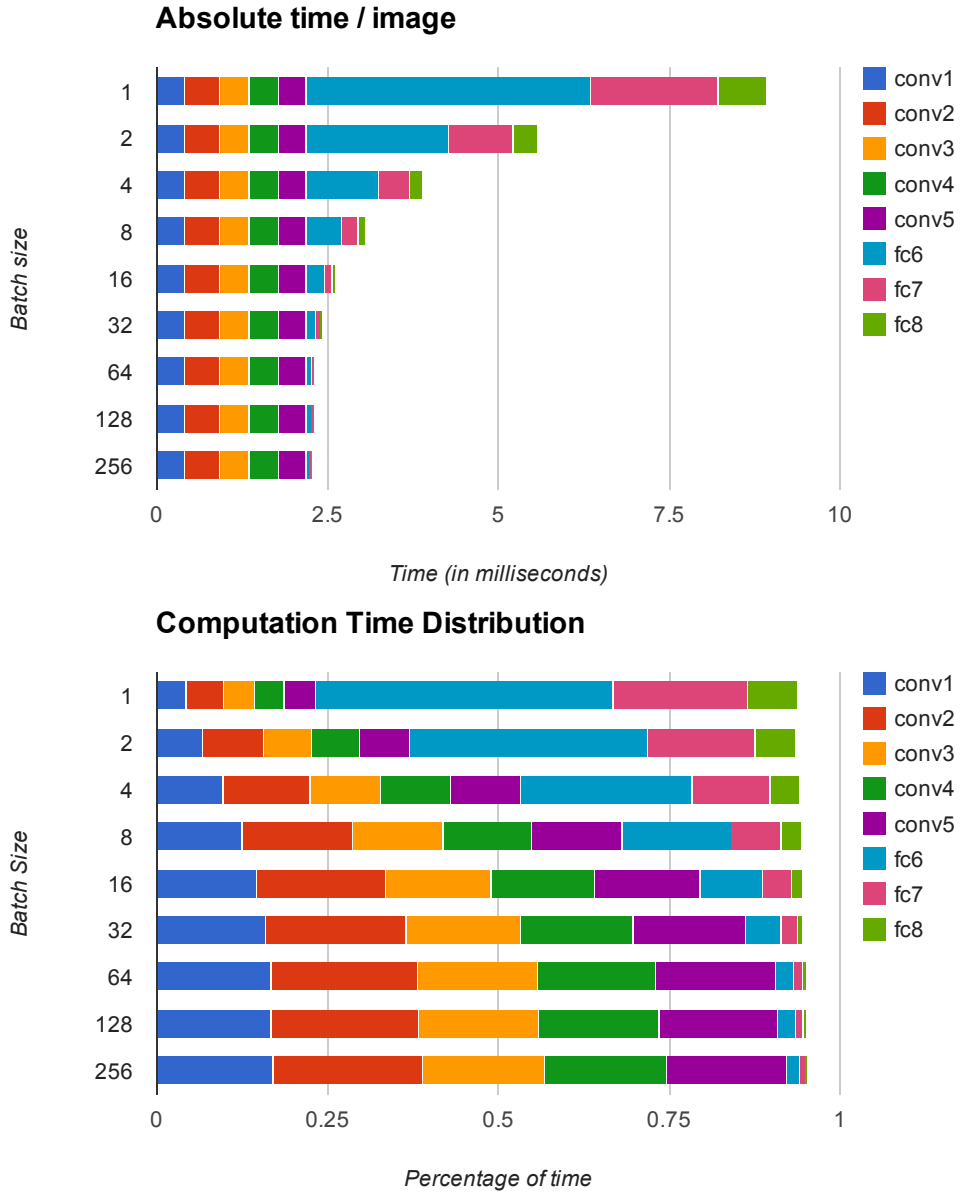


Figure 6.2: Top: absolute computation time spent on each layer per image, when the batch size varies from 1 to 256. Bottom: an alternative view of the top chart, showing the relative percentage of computation time by each layer. Note that in both figures, low-cost layers such as pooling and ReLU are not shown, which accounts for the 5% unoccupied percentage in the bottom figure.

6.4 On the Effectiveness of Feature Transfer

Since the network is trained with a large number of images and a large number of classes, a reasonable conjecture is that intermediate representations in this network will be useful as a general-purpose feature that performs well on related tasks, in a feature transfer fashion. In this section, a systematic examination is carried out to verify to what extent such conjecture holds.

Specifically, we present experimental results on multiple standard computer vision benchmarks, comparing many possible featurization and classification approaches. In each of the experiments, we take the activations of the n^{th} hidden layer of the deep convolutional neural network described in the previous section as a feature. The following layers are used for evaluation: (1) FC_7, which denotes features taken from the final hidden layer – i.e., just before propagating through the final fully connected layer to produce the class predictions; (2) FC_6, which is the activations of the layer before FC_7, and (3) POOL_5, which is the last set of activations that has been fully propagated through the convolutional layers of the network. We chose not to evaluate features from any earlier in the network, as the earlier convolutional layers are unlikely to contain a richer semantic representation than the later features which form higher-level hypotheses from the low to mid-level local information in the activations of the convolutional layers. Because we are investigating the use of the network's hidden layer activations as features, all of its weights are frozen to those learned on the ILSVRC2012 dataset.[7] All images are preprocessed using the same procedure as in ILSVRC: taking features on the center 227×227 crop of the 256×256 resized image.

We present results on multiple datasets to evaluate the strength of Caffe for basic object recognition, domain adaptation, fine-grained recognition, and scene recognition. These tasks each differ somewhat from that for which the architecture was trained, together representing much of the contemporary visual recognition spectrum.

6.4.1 General Object Recognition

To analyze the ability of the deep features to transfer to basic-level object category recognition, we evaluate them on the Caltech-101 dataset [36]. In addition to directly evaluating linear classifier performance on FC_6 and FC_7, we also report results using the dropout regularization technique proposed by [49], which is also present in the Imagenet training phase. At training time, this technique works by randomly setting half of the activations (here, our features) in a given layer to 0. At test time, all activations are multiplied by 0.5. Dropout was used successfully by [65] in layers 6 and 7 of their network; hence we study the effect of the technique when applied to the features derived from these layers.

In each evaluation, the classifier, a logistic regression (LogReg) or support vector machine (SVM), is trained on a random set of 30 samples per class (including the background class), and tested on the rest of the data, with parameters cross-validated for each split on a 25

[7] We also experimented with the equivalent feature using randomized weights and found it to have performance comparable to traditional hand-designed features.

	$POOL_5$	FC_6	FC_7
LogReg	63.29 ± 6.6	84.30 ± 1.6	84.87 ± 0.6
LogReg with Dropout	-	86.08 ± 0.8	85.68 ± 0.6
SVM	77.12 ± 1.1	84.77 ± 1.2	83.24 ± 1.2
SVM with Dropout	-	$\mathbf{86.91 \pm 0.7}$	85.51 ± 0.9
Yang *et al.* [119]		84.3	
Jarrett *et al.* [53]		65.5	

Table 6.2: Average accuracy per class on Caltech-101 with 30 training samples per class across three hidden layers of the network and two classifiers. Our result from the training protocol/classifier combination with the best validation accuracy – SVM with Layer 6 (+ dropout) features – is shown in **bold**.

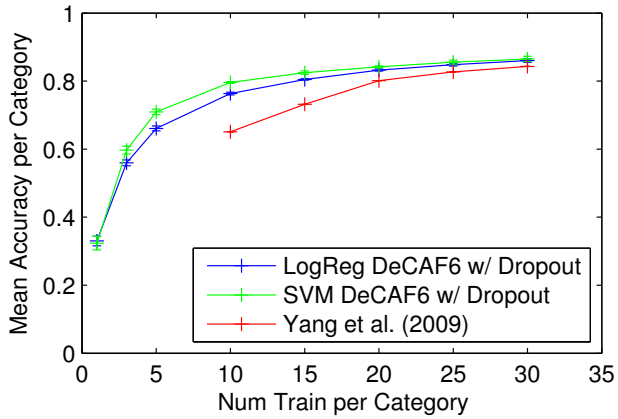


Figure 6.3: Average accuracy per class on Caltech-101 at varying training set sizes.

train/5 validation sub-split of the training data. The results in Table 6.2 are reported in terms of mean accuracy per category averaged over five data splits.

Our top-performing method (based on validation accuracy) trains a linear SVM on FC_6 with dropout, with test set accuracy of 86.9%. The $POOL_5$ features perform substantially worse than either the FC_6 or FC_7 features, and hence we do not evaluate them further in this work. The FC_7 features generally have accuracy about 1-2% lower than the FC_6 features on this task. The dropout regularization technique uniformly improved results by 0-2% for each classifier/feature combination. When trained on the deep features, the SVM and logistic regression classifiers perform roughly equally well on this task.

We compare our performance against the current state-of-the-art on this benchmark from [119], a method employing a combination of 5 traditional hand-engineered image features followed by a multi-kernel based classifier. Our top-performing method outperforms this method by 2.6%. Note that it is likely that our features could be added to the set of hand-engineered features inside of the method of [119] to achieve even higher performance,

but we focus here only on very simple, fast methods to demonstrate the representational strength of the features alone. Our method also outperforms by over 20% the two-layer convolutional network of [53], demonstrating the importance of the depth of the network used for our feature. Note that unlike our method, these approaches from the literature do not implicitly leverage an outside large-scale image database like ImageNet. The performance edge of our method over these approaches demonstrates the importance of multi-task learning when performing object recognition with sparse data like that available in the Caltech-101 benchmark.

To further explore the impact of sparse training data on our method's performance, we also show how performance of the two Layer 6 + dropout methods above vary with the number of training cases per category, plotted in Figure 6.3. Results are again given in terms of mean accuracy per class averaged across five random data splits, and parameters in this case were kept fixed. The i^{th}training set split at size M is a subset of the i^{th} split at size N for any $N > M$ to ensure consistency between results. Surprisingly, with just a *single* labeled instance per category (one-shot learning) we are able to learn a reasonably useful classifier on the 102-category classification task, achieving accuracy of 33.0% compared to chance accuracy of $< 1\%$. This suggest that with sufficiently strong representations like the ones in this thesis, useful models of visual categories can often be learned from just a single positive example.

6.4.2 Domain Adaptation

We next evaluate the deep features for use on the task of domain adaptation. For our experiments we use the benchmark *Office* dataset [95].The dataset contains three domains: `Amazon`, which consists of product images taken from `amazon.com`; and `Webcam` and `Dslr`, which consist of images taken in an office environment using a webcam or digital SLR camera, respectively.

In the domain adaptation setting, we are given a training (source) domain with labeled training data and a distinct test (target) domain with either a small amount of labeled data or no labeled data. We will experiment within the supervised domain adaptation setting, where there is a small amount of labeled data available from the target domain.

Most prior work for this dataset uses SURF [4] interest point features (available for download with the dataset). To illustrate the ability of deep features to be robust to resolution changes, we use the t-SNE [79] algorithm to project both SURF and FC_6, computed for `Webcam` and `Dslr`, into a 2D visualizable space (See Figure 6.4). We visualize an image on the point in space corresponding to its low dimension projected feature vector. We find that the deep features not only provides better within category clustering, but also clusters same category instances across domains, effectively *undoing* the domain bias, or the "dataset bias" as described by Torralba and Efros [110].

We validate this conclusion with a quantitative experiment on the *Office* dataset. Table 6.3 presents multi-class accuracy averaged across 5 train/test splits for the domain shifts `Amazon`→`Webcam` and `Dslr` → `Webcam`. We use the standard experimental setup first pre-

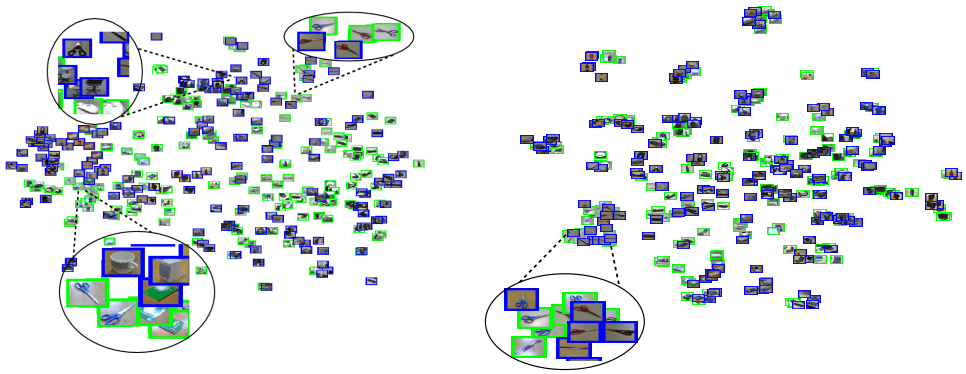
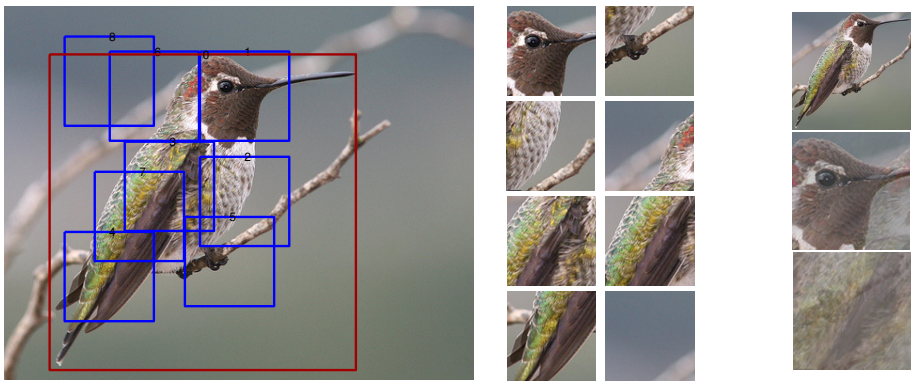


Figure 6.4: Visualization of the webcam (green) and dslr (blue) domains using the original released SURF features (left) and FC_6 (right). The figure is best viewed by zooming in to see the images in local regions. All images from the scissor class are shown enlarged. They are well clustered and overlapping in both domains with our representation, while SURF only clusters a subset and places the others in disjoint parts of the space, closest to distinctly different categories such as chairs and mugs.

	`Amazon` → `Webcam`			`Dslr` → `Webcam`		
	SURF	FC_6	FC_7	SURF	FC_6	FC_7
LogReg(S)	9.63 ± 1.4	48.58 ± 1.3	53.56 ± 1.5	24.22 ± 1.8	88.77 ± 1.2	87.38 ± 2.2
SVM(S)	11.05 ± 2.3	52.22 ± 1.7	53.90 ± 2.2	38.80 ± 0.7	91.48 ± 1.5	89.15 ± 1.7
LogReg(T)	24.33 ± 2.1	72.56 ± 2.1	74.19 ± 2.8	24.33 ± 2.1	72.56 ± 2.1	74.19 ± 2.8
SVM(T)	51.05 ± 2.0	78.26 ± 2.6	78.72 ± 2.3	51.05 ± 2.0	78.26 ± 2.6	78.72 ± 2.3
LogReg(ST)	19.89 ± 1.7	75.30 ± 2.0	76.32 ± 2.0	36.55 ± 2.2	92.88 ± 0.6	91.91 ± 2.0
SVM(ST)	23.19 ± 3.5	80.66 ± 2.3	79.12 ± 2.1	46.32 ± 1.1	$\mathbf{m}94.79 \pm 1.2$	92.96 ± 2.0
[26]	40.26 ± 1.1	$\mathbf{m}82.14 \pm 1.9$	81.65 ± 2.4	55.07 ± 3.0	91.25 ± 1.1	89.52 ± 2.2
[50]	37.66 ± 2.2	80.06 ± 2.7	80.37 ± 2.0	53.65 ± 3.3	93.25 ± 1.5	91.45 ± 1.5
[45]	39.80 ± 2.3	75.21 ± 1.2	77.55 ± 1.9	39.12 ± 1.3	88.40 ± 1.0	88.66 ± 1.9
[16]		58.85			78.21	

Table 6.3: Deep features dramatically outperforms the baseline SURF feature available with the *Office* dataset as well as the deep adaptive method of [16]. We report average multi-class accuracy using both standard and adaptive classifiers, changing only the input feature from SURF to deep features. Surprisingly, in the case of `Dslr`→`Webcam` the domain shift is largely non-existent with the new features.



DPM Detections Parts DPD

Figure 6.5: The pipeline of the deformable part descriptor (DPD) on a sample test images. It uses DPM for part localization and then uses learned pooling weights for the final pose-normalized representation.

Method	Accuracy
FC_6	58.75
DPD + FC_6	**64.96**
DPD [124]	50.98
POOF [7]	56.78

Table 6.4: Accuracy on the Caltech-UCSD bird dataset.

sented in [95]. To compare SURF with the FC_6 and FC_7 features, we report the multi-class accuracy for each, using an SVM and Logistic Regression both trained in 3 ways: with only source data (S), only target data (T), and source and target data (ST). We also report results for three adaptive methods run with each deep feature we consider as input. Finally, for completeness we report a recent and competing deep domain adaptation result from [16]. It is observed that by adopting deep features alone, one could dramatically outperform the baseline SURF feature available with the *Office* dataset as well as the deep adaptive method of [16].

6.4.3 Fine-Grained Recognition

We tested the performance of deep features on the task of subcategory recognition. To this end, we adopted one of its most popular tasks - the Caltech-UCSD birds dataset [113], and compare the performance against several state-of-the-art baselines.

Following common practice in the literature, we adopted two approaches to perform classification. Our first approach adopts an ImageNet-like pipeline, in which we followed

the existing protocol by cropping the images regions 1.5× the size of the provided bounding boxes, resizing them 256×256 and then feeding them into the convolutional pipeline to get the features for classification. For simplicity, we computed FC_6 and trained a multi-class logistic regression on top of the features.

Our second approach, we tested the deep features in a pose-normalized setting using the deformable part descriptors (DPD) method [124]. Inspired by the deformable parts model [39], DPD explicitly utilizes the part localization to do semantic pooling. Specifically, after training a weakly-supervised DPM on bird images, the pool weight for each part of each component is calculated by using the key-point annotations to get cross-component semantic part correspondence. The final pose-normalized representation is computed by pooling the image features of predicted part boxes using the pooling weights. Based on the DPD implementation provided by the authors, we applied deep feature extraction in the same pre-trained DPM model and part predictions and used the same pooling weights. Figure 6.5 shows the DPM detections and visualization of pooled DPD features on a sample test image. As our first approach, we resized each predicted part box to 256 × 256 and computed FC_6 to replace the KDES image features [8] used in the original DPD paper [124].

Our performance as well as those from the literature are listed in Table 6.4. Deep features together with a simple logistic regression already obtains a significant performance increase over existing approaches, indicating that such features, although not specifically designed to model subcategory-level differences, captures such information well. In addition, explicitly taking more structured information such as part locations still helps, and provides another significant performance increase, obtaining an accuracy of 64.96%, compared to the 50.98% accuracy reported in [124]. It also outperforms POOF [7], to our knowledge the best accuracy reported in the literature prior to this work.

We note again that in all the experiments above, no fine-tuning is carried out on the lower-level layers, since our main interest is to analyze how well a feature extraction pipeline trained with a different objective generalizes to different tasks. To obtain the best possible result one may want to perform a full back-propagation, as have been shown effective in specific applications like detection [44]. However, the fact that we see a significant performance increase without fine-tuning suggests that a pre-trained deep network may already serve as a good off-the-shelf visual representation without heavy computation.

6.4.4 Scene Recognition

Finally, we evaluate the deep features on the scene recognition benchmarks, namely the SUN-397 large-scale scene recognition database [115]. Unlike object recognition, wherein the goal is to identify and classify an object which is usually the primary focus of the image, the goal of a scene recognition task is to classify the *scene* of the entire image. In the SUN-397 database, there are 397 semantic scene categories including *abbey*, *diner*, *mosque*, and *stadium*. Because the Caffe features are learned on ILSVRC, an object recognition database, we are applying it to a task for which it was not designed. Hence we might expect this task

	FC_6	FC_7
LogReg	$\mathbf{m}40.94 \pm 0.3$	40.84 ± 0.3
SVM	39.36 ± 0.3	40.66 ± 0.3
Xiao *et al.* [115]	38.0	

Table 6.5: Average accuracy per class on SUN-397 with 50 training samples and 50 test samples per class, across two hidden layers of the network and two classifiers. Note that the result from the training protocol/classifier combination with the best validation accuracy is FC_7, while FC_6 has the best testing accuracy. However the difference between them is not statistically significant.

to be very challenging for these features, unless they are highly generic representations of the visual world.

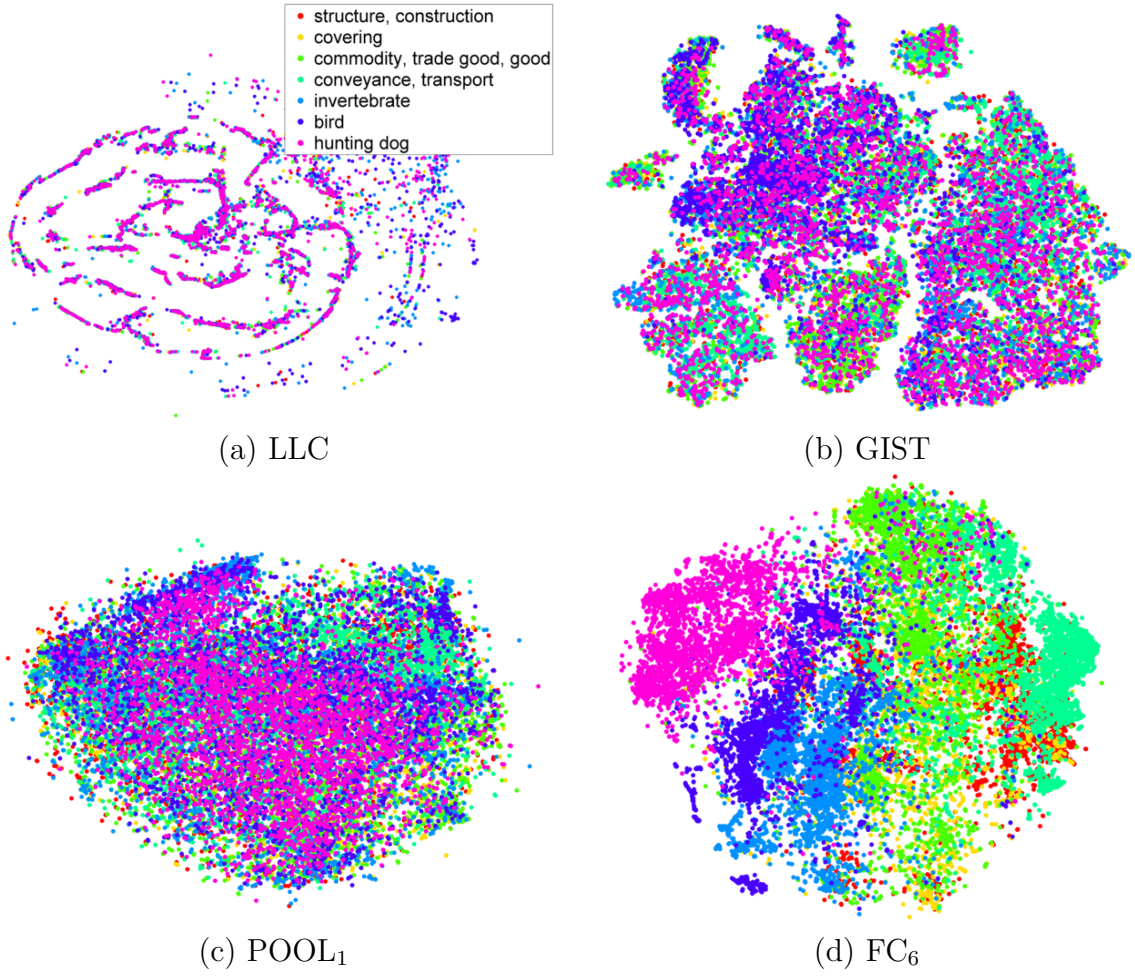
Based on the success of using dropout with FC_6 and FC_7 for the object recognition task detailed in Section 6.4.1, we train and evaluate linear classifiers on these dropped-out features on the SUN-397 database. Table 6.5 gives the classification accuracy results averaged across 5 splits of 50 training images and 50 test images. Parameters are fixed for all methods, but we select the top-performing method by cross-validation, training on 42 images and testing on the remaining 8 in each split.

Our top-performing method in terms of cross-validation accuracy was to use FC_7 with the SVM classifier, resulting in 40.94% test performance. Comparing against the method of [115], the current state-of-the-art method, we see a performance improvement of 2.9% using the feature off-the-shelf and without any ensembles with additional features. Note that, like the state-of-the-art method used as a baseline in Section 6.4.1, this method uses a large set of traditional vision features and combines them with a multi-kernel learning method. The fact that a simple linear classifier on top of our single image feature outperforms the multi-kernel learning baseline built on top of many traditional features demonstrates the ability of deep features to generalize to other tasks and its representational power as compared to traditional hand-engineered features.

6.5 Emergence of Conceptual Embeddings

In addition to quantitatively analyze the learned features by stacking a classifier on top of them, an alternate way to qualitatively show the effectiveness of various features is to look at their distribution in the feature space with respect to high-level semantics, such as object labels. To this end, we will visualized the model features to gain insight into the semantic capacity of the various features that have been typically employed in computer vision. In particular, we compare the learned deep features with GIST features [89] and LLC features [112].

The features in the following way: we run the t-SNE algorithm [79] to find a 2-dimensional embedding of the high-dimensional feature space, and plot them as points colored depending



(a) LLC (b) GIST

(c) $POOL_1$ (d) FC_6

Figure 6.6: This figure shows several t-SNE feature visualizations on the ILSVRC-2012 validation set. (a) LLC , (b) GIST, and features derived from our CNN: (c) $POOL_1$, the first pooling layer, and (d) FC_6, the second to last hidden layer. The figure is best viewed in color.

on their semantic category in a particular hierarchy. We did this on the validation set of ILSVRC-2012 to avoid overfitting effects (as the deep CNN used in this work was trained only on the training set), and also use an independent dataset, SUN-397 [115], to evaluate how dataset bias affects our results (see e.g. [110] for a deeper discussion of this topic). For high-dimensional features such as LLC, we first generate a random orthonormal matrix and perform dimensionality reduction to obtain a 512-dimensional feature space. It is known from the JohnsonLindenstrauss lemma that such random projection preserves Euclidean distance to a satisfying granularity, thus not hurting the final embedding result.

We first visualize the semantic segregation of the model by plotting the embedding of labels for higher levels of the WordNet hierarchy; for example, a strong feature for visual

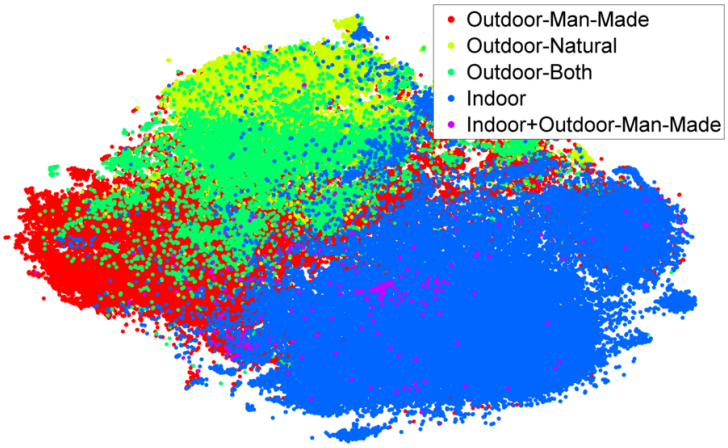


Figure 6.7: In this figure we show how our features trained on ILSVRC-2012 generalized to SUN-397 when considering semantic groupings of labels. Best viewed in color.

recognition should cluster animals and non-animals instances separately, even though there is no explicit modeling through the supervised training of the CNN[8]. Figure 6.6 shows the features extracted on the validation set using the first pooling layer, and the second to last hidden layer FC_6, showing a clear semantic clustering in the latter but not in the former. This is compatible with common deep learning knowledge that the first layers learn "low-level" features, whereas the latter layers learn semantic or "high-level" features.

Furthermore, baseline features such as GIST or LLC fail to capture the semantic difference in the image (although they show interesting clustering structure). For GIST, it focuses more on holistic layouts of the objects, but does not take into account specific label information. For LLC, the embedding is highly sparse, possibly due to the fact that LLC is a highly over-complete feature designed to work well with linear kernels, effectively pulling all data points far from each other.

More interestingly, we note that such semantic segregation is, at least to some extent, transferrable to related but different datasets. In Figure 6.7 we visualize the top performing features (FC_6) on the SUN-397 dataset, also embedded with t-SNE in a two dimensional space. Although the features are never trained to distinguish scenes and have not seen such labels, they show very good clustering of semantic classes (e.g., indoor vs. outdoor). This suggests that intermediate layer outputs serve as good features for general object recognition tasks, considering the case where the object class that we are trying to detect is not in the original object pool of ILSVRC-2012.

[8]We note that during training, labels are considered flat, *i.e.,* "cat" is as negative as "car" for a dog image. Thus, the semantic segregation of high-level clusters should be interpreted as emerging from data, not supervised information.

6.5.1 Perceptual Embedding of Visual Concepts: A Conjecture

It is worth noting that the natural emergence of concept-level grouping is closely related to the concept learning framework I proposed in the previous chapter. Specifically, existing concept learning frameworks requires a pre-designed hierarchy, which defines the "closeness" between various object categories in a conceptual space, to be provided as oracle knowledge. While this is available in specific use cases such as WordNet, it is nonetheless complete, and determining the correct conceptual hierarchy is an open problem yet to be explored.

With the intermediate features from a deep model, here I present a conjecture that grounds semantics directly in the *perceptual* space obtained from deep features, rather than a separate label hierarchy that is manually constructed. Specifically, one could view the distance in the embedded space as a measure of semantic closeness. For example, for an image of dalmatian, closest to it will be other related dalmatians; further in the space would be other breeds of dogs, other animals, and finally non-animal images. Two possible directions would naturally follow:

Learning Nested Embeddings What has been shown in the previous section is the natural emergence of conceptual hierarchy in the flat learning procedure. While this has already been performing well qualitatively, a natural choice to learn a better embedding space is to utilize the hierarchy that ImageNet/WordNet provides. Specifically, one would be able to learn a better embedding with what could be called "nested distance metric learning": for a triplet of images such as a dalmatian, a corgi and a tabby cat, one can enforce relative comparison constraints [100].

Learning Concept Prior With the learned embedded space, one could construct concept priors and conditionals in the perceptual space following the similar ideas we use for the conceptual space. One possible concept definition that is most likely to succeed is to define concepts as Gaussian distributions[9] in the perceptual space, with the standard deviation of the distribution serving as the "width" of the concept: for example, "dog" would have a larger width than "dalmatian". The prior of such concepts could then be constructed on top of the mean and the variance of such gaussian distributions, and a mathematically convenient choice would be a Wishart distribution, leading to a conjugate prior-conditional pair.

Interestingly enough, a very similar idea has been proposed by Fei-Fei *et al.* [37], although it was only proposed and evaluated in a limited fashion on the Caltech-101 dataset with conventional, hand-crafted features. A more detailed mathematical analysis as well as large-scale behavioral experiments will be one of my research directions in the future.

[9] I propose Gaussian distribution to provide some level of continuity, although one could certainly binarize the probability to obtain 0-1 predictions.

6.6 Summary

This chapter serves as the most exploratory chapter of the whole thesis: I presented Caffe, the open-source deep learning library that allows various research and applications to be built upon. Based on it, I have presented extensive analysis on how such deep features behave in various tasks, and propose possible direction towards the main topic of this thesis - to eventually learn a perceptual embedded space for concept hierarchies.

As a final note, the Caffe package was released in December 2013 and is now under the 2-clause BSD license. More information could be found at `http://caffe.berkeleyvision.org/`.

Chapter 7

Conclusion

In this thesis I have analyzed two key components in the computer vision research: to learn better image features with solid theoretical justifications, and to re-visit the existing vision problem statement to a more practical and human-like one. To this end, I have proposed and analyzed novel receptive field learning and dictionary learning methods, which is justified by the Nyström sampling theory, that learns more compact and effective features for object recognition tasks. Having analyzed the feature generation pipeline, I then move to the more high-level scrutiny of the current object recognition experimental setting. By combining the otherwise independently developing computer vision and cognitive science studies, I have presented the first large-scale system that allows computers to learn and generalize closer to what a human learner will do. I have also collected a large-scale human behavior database, which would hopefully enable further research along this research direction.

It is both surprising and thrilling to witness the recent success, or rediscovery as we may argue, of convolutional neural networks in the object recognition research. I have devoted the last part of my thesis to making a better engineering system for deep learning research, as well as providing an extensive analysis on how features learned from the state-of-the-art CNN pipeline would serve as a general-purpose visual descriptor that could be adopted in various applications. Computer vision research has always been pushed by great open-source vision libraries, and it is my sincere hope that my contribution would boost this new field in the coming years.

Bibliography

[1] JT Abbott, JL Austerweil, and TL Griffiths. "Constructing a hypothesis space from the Web for large-scale Bayesian word learning". In: *Proceedings of the 34th Annual Conference of the Cognitive Science Society*. 2012.

[2] R Ando and T Zhang. "A framework for learning predictive structures from multiple tasks and unlabeled data". In: *JMLR* 6 (2005).

[3] A Argyriou, T Evgeniou, and M Pontil. "Multi-Task Feature Learning". In: *NIPS*. 2006.

[4] H Bay, T Tuytelaars, and L Van Gool. "SURF: Speeded Up Robust Features". In: *ECCV*. 2006.

[5] A Berg, J Deng, and L Fei-Fei. *ILSVRC 2010*. `http://www.image-net.org/challenges/LSVRC/2010/`.

[6] A Berg, J Deng, and L Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge 2012". In: (2012). URL: `http://www.image-net.org/challenges/LSVRC/2012/`.

[7] T Berg and P Belhumeur. "POOF: Part-Based One-vs-One Features for Fine-Grained Categorization, Face Verification, and Attribute Estimation". In: *CVPR*. 2013.

[8] L Bo, X Ren, and D Fox. "Kernel Descriptors for Visual Recognition". In: *NIPS*. 2010.

[9] O Boiman, E Shechtman, and M Irani. "In defense of nearest-neighbor based image classification". In: *CVPR*. 2008.

[10] L Bourdev, S Maji, and J Malik. "Describing People: A Poselet-Based Approach to Attribute Classification". In: *ICCV*. 2011.

[11] Y Boureau and J Ponce. "A theoretical analysis of feature pooling in visual recognition". In: *ICML*. 2010.

[12] Y Boureau et al. "Ask the locals: multi-way local pooling for image recognition". In: *ICCV*. 2011.

[13] Y Boureau et al. "Learning mid-level features for recognition". In: *CVPR*. 2010.

[14] S Carey. "The child as word learner". In: *Linguistic Theory and Psychological Reality* (1978).

[15] R Caruana. "Multitask Learning". In: *Machine Learning* 28 (1997).

[16] S Chopra, S Balakrishnan, and R Gopalan. "DLID: Deep Learning for Domain Adaptation by Interpolating between Domains". In: *ICML Workshop on Challenges in Representation Learning*. 2013.

[17] DC Cireşan et al. "High-performance neural networks for visual object classification". In: *ArXiv e-prints* (2011). arXiv: 1102.0183.

[18] A Coates, A Karpathy, and AY Ng. "Emergence of Object-Selective Features in Unsupervised Feature Learning". In: *NIPS*. 2012.

[19] A Coates, H Lee, and AY Ng. "An analysis of single-layer networks in unsupervised feature learning". In: *AISTATS*. 2010.

[20] A Coates and A Ng. "The importance of encoding versus training with sparse coding and vector quantization". In: *ICML*. 2011.

[21] A Coates and AY Ng. "Selecting Receptive Fields in Deep Networks". In: *NIPS*. 2011.

[22] A Coates et al. "Deep learning with COTS HPC systems". In: *Proceedings of The 30th International Conference on Machine Learning*. 2013, pp. 1337–1345.

[23] C Cortes, M Mohri, and A Talwalkar. "On the Impact of Kernel Approximation on Learning Accuracy". In: *AISTATS*. 2010.

[24] N Dalal. "Histograms of oriented gradients for human detection". In: *CVPR*. 2005.

[25] N Dalal and B Triggs. "Histograms of oriented gradients for human detection". In: *CVPR*. 2005.

[26] H Daume III. "Frustratingly Easy Domain Adaptation". In: *ACL*. 2007.

[27] J Dean et al. "Large Scale Distributed Deep Networks". In: *NIPS*. 2012.

[28] J Deng et al. "Hedging Your Bets: Optimizing Accuracy-Specificity Trade-offs in Large Scale Visual Recognition". In: *CVPR*. 2012.

[29] J Deng et al. "ImageNet: A large-scale hierarchical image database". In: *CVPR*. 2009.

[30] M Denil and N de Freitas. "Recklessly Approximate Sparse Coding". In: *arXiv preprint arXiv:1208.0959* (2012).

[31] J Donahue et al. "Decaf: A deep convolutional activation feature for generic visual recognition". In: *arXiv preprint arXiv:1310.1531* (2013).

[32] J Duchi, E Hazan, and Y Singer. "Adaptive subgradient methods for online learning and stochastic optimization". In: *JMLR* 12 (2010), pp. 2121–2159.

[33] M Everingham et al. "The PASCAL Visual Object Classes (VOC) Challenge". In: *IJCV* 88.2 (2010), pp. 303–338.

[34] A Farhadi et al. "Describing objects by their attributes". In: *CVPR*. 2009.

[35] R Farrell et al. "Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance". In: *CVPR*. 2011.

[36] L Fei-Fei, R Fergus, and P Perona. "Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories". In: *CVPR*. 2004.

[37] L Fei-Fei, R Fergus, and P Perona. "One-shot learning of object categories". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 28.4 (2006), pp. 594–611.

[38] C Fellbaum. "WordNet". In: *Theory and Applications of Ontology: Computer Applications* (2010), pp. 231–243.

[39] P Felzenszwalb et al. "Object Detection with Discriminatively Trained Part-Based Models". In: *PAMI* 32 (2010).

[40] J Feng et al. "Geometric Lp-norm Feature Pooling for Image Classification". In: *CVPR*. 2011.

[41] S Fidler and A Leonardis. "Towards Scalable Representations of Object Categories: Learning a Hierarchy of Parts". In: *CVPR*. 2007.

[42] BJ Frey and D Dueck. "Clustering by passing messages between data points". In: *Science* 315.5814 (2007), pp. 972–976.

[43] T Gao and D Koller. "Discriminative learning of relaxed hierarchy for large-scale visual recognition". In: *ICCV*. 2011.

[44] R Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *arXiv preprint arXiv:1311.2524* (2013).

[45] B Gong et al. "Geodesic Flow Kernel for Unsupervised Domain Adaptation". In: *CVPR*. 2012.

[46] LK Hansen and J Larsen. "Linear unlearning for cross-validation". In: *Advances in Computational Mathematics* 5.1 (1996), pp. 269–280.

[47] Z Harchaoui et al. "Large-scale image classification with trace-norm regularization". In: *CVPR*. 2012.

[48] G Hinton and R Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science* (2006).

[49] G Hinton et al. "Improving Neural Networks by Preventing Co-adaptation of Feature Detectors". In: *arXiv preprint arXiv:1207.0580* (2012).

[50] J Hoffman et al. "Efficient Learning of Domain-invariant Image Representations". In: *ICLR*. 2013.

[51] DH Hubel and TN Wiesel. "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex". In: *The Journal of Physiology* 160 (1962), pp. 106–154.

[52] L Itti and C Koch. "Computational modeling of visual attention". In: *Nature reviews neuroscience* (2001).

[53] K Jarrett et al. "What is the Best Multi-Stage Architecture for Object Recognition?" In: *ICCV*. 2009.

[54] R Jenatton, G Obozinski, and F Bach. "Structured sparse principal component analysis". In: *AISTATS*. 2010.

[55] Y Jia and T Darrell. "Latent Task Adaptation with Large-scale Hierarchies". In: *ICCV*. 2013.

[56] Y Jia, C Huang, and T Darrell. "Beyond spatial pyramids: Receptive field learning for pooled image features". In: *CVPR*. 2012.

[57] Y Jia, O Vinyals, and T Darrell. "On compact codes for spatially pooled features". In: *ICML*. 2013.

[58] Y Jia et al. "Visual Concept Learning: Combining Machine Vision and Bayesian Generalization on Concept Hierarchies". In: *NIPS*. 2013.

[59] D Jurafsky and JH Martin. *Speech & Language Processing*. Pearson Prentice Hall, 2000.

[60] S Karayev et al. "A Probabilistic Model for Recursive Factorized Image Features". In: *CVPR*. 2011.

[61] L Kennedy and A Hauptmann. "LSCOM Lexicon Definitions and Annotations (Version 1.0)". In: (2006).

[62] A Khosla et al. "Novel dataset for fine-grained image categorization". In: *CVPR FGVC workshop*. 2011.

[63] JJ Koenderink and AJ van Doorn. "The structure of locally orderless images". In: *IJCV* 31.2/3 (1999), pp. 159–168.

[64] A Krizhevsky. "Convolutional deep belief networks on CIFAR-10". In: *Technical Report* (2010).

[65] A Krizhevsky, I Sutskever, and GE Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *NIPS*. 2012.

[66] B Kulis, K Saenko, and T Darrell. "What you saw is not what you get: Domain adaptation using asymmetric kernel transforms". In: *CVPR*. 2011.

[67] S Kumar, M Mohri, and A Talwalkar. "Sampling methods for the Nyström method". In: *JMLR* 13.Apr (2012), pp. 981–1006.

[68] K Labusch, E Barth, and T Martinetz. "Simple method for high-performance digit recognition based on sparse coding". In: *IEEE TNN* 19.11 (2008), pp. 1985–1989.

[69] F Lauer, CY Suen, and G Bloch. "A trainable feature extractor for handwritten digit recognition". In: *Pattern Recognition* 40.6 (2007), pp. 1816–1824.

[70] S Lazebnik, C Schmid, and J Ponce. "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories". In: *CVPR*. 2006.

[71] Q Le et al. "Building high-level features using large scale unsupervised learning". In: *ICML*. 2012.

[72] Q Le et al. "Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis". In: *CVPR*. 2011.

[73] Y LeCun et al. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* (1989).

[74] Y LeCun et al. "Gradient-based learning applied to document recognition". In: *Proc. of the IEEE* 86.11 (1998), pp. 2278–2324.

[75] L Li et al. "Object bank: A high-level image representation for scene classification & semantic feature sparsification". In: *NIPS*. 2010.

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

[77] Y Lin et al. "Large-scale image classification: fast feature extraction and svm training". In: *CVPR*. 2011.

[78] D Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *IJCV* 60 (2004).

[79] L van der Maaten and GE Hinton. "Visualizing Data using t-SNE". In: *JMLR* 9 (2008).

[80] J Mairal et al. "Online learning for matrix factorization and sparse coding". In: *JMLR* 11 (2010), pp. 19–60.

[81] EM Markman. *Categorization and naming in children: Problems of induction*. MIT Press, 1991.

[82] G Mesnil et al. "Unsupervised and Transfer Learning Challenge: a Deep Learning Approach." In: *JMLR* 27 (2012).

[83] CD Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, 2001.

[84] A Oliva and A Torralba. "Modeling the shape of the scene: A holistic representation of the spatial envelope". In: *International journal of computer vision* 42.3 (2001), pp. 145–175.

[85] B Olshausen and DJ Field. "Sparse coding with an overcomplete basis set: a strategy employed by V1?" In: *Vision research* 37.23 (1997), pp. 3311–3325.

[86] D Parikh and K Grauman. "Relative attributes". In: *ICCV*. 2011.

[87] S Perkins, K Lacker, and J Theiler. "Grafting: fast, incremental feature selection by gradient descent in function space". In: *JMLR* 3 (2003), pp. 1333–1356.

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

[89] A Quattoni and A Torralba. "Recognizing indoor scenes". In: *CVPR*. 2009.

[90] R Raina et al. "Self-taught learning: transfer learning from unlabeled data". In: *ICML*. 2007.

[91] MA Ranzato et al. "Unsupervised learning of invariant feature hierarchies with applications to object recognition". In: *CVPR*. 2007.

[92] X Ren and D Ramanan. "Histograms of Sparse Codes for Object Detection". In: *CVPR*. 2013.

[93] R Rigamonti, MA Brown, and V Lepetit. "Are sparse representations really relevant for image classification?" In: *CVPR*. 2011.

[94] B Russell et al. "LabelMe: a database and web-based tool for image annotation". In: *International journal of computer vision* 77.1-3 (2008), pp. 157–173.

[95] K Saenko et al. "Adapting visual category models to new domains". In: *ECCV*. 2010.

[96] R Salakhutdinov, A Torralba, and JB Tenenbaum. "Learning to share visual appearance for multiclass object detection". In: *CVPR*. 2011.

[97] J Sánchez and F Perronnin. "High-dimensional signature compression for large-scale image classification". In: *CVPR*. 2011.

[98] A Saxe et al. "On random weights and unsupervised feature learning". In: *ICML*. 2011.

[99] M Schmidt et al. "Structure learning in random fields for heart motion abnormality detection". In: *CVPR*. 2008.

[100] M Schultz and T Joachims. "Learning a Distance Metric from Relative Comparisons." In: *NIPS*. 2003.

[101] RN Shepard. "Toward a universal law of generalization for psychological science". In: *Science* 237.4820 (1987), pp. 1317–1323.

[102] S Singh, A Gupta, and A Efros. "Unsupervised Discovery of Mid-Level Discriminative Patches". In: *ECCV*. 2012.

[103] Ameet Talwalkar and Afshin Rostamizadeh. "Matrix coherence and the nystrom method". In: *arXiv preprint arXiv:1004.2008* (2010).

[104] JB Tenenbaum. "Bayesian modeling of human concept learning". In: *NIPS*. 1999.

[105] JB Tenenbaum and TL Griffiths. "Generalization, similarity, and Bayesian inference". In: *Behavioral and Brain Sciences* 24.4 (2001), pp. 629–640.

[106] Joshua B Tenenbaum, Thomas L Griffiths, Charles Kemp, et al. "Theory-based Bayesian models of inductive learning and reasoning". In: *Trends in cognitive sciences* 10.7 (2006), pp. 309–318.

[107] S Thrun. "Is Learning the n-th Thing any Easier Than Learning the First?" In: *NIPS*. 1996.

[108] R Tibshirani. "Regression shrinkage and selection via the lasso". In: *JRSS Series B* (1996), pp. 267–288.

[109] A Torralba. "Contextual priming for object detection". In: *IJCV* 53.2 (2003), pp. 169–191.

[110] A Torralba and A Efros. "Unbiased Look at Dataset Bias". In: *CVPR*. 2011.

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

[112] J Wang et al. "Locality-constrained linear coding for image classification". In: *CVPR*. 2010.

[113] P Welinder et al. *Caltech-UCSD Birds 200*. Tech. rep. CNS-TR-2010-001. California Institute of Technology, 2010.

[114] S Winder and M Brown. "Learning local image descriptors". In: *CVPR*. 2007.

[115] J Xiao et al. "SUN Database: Large-scale Scene Recognition from Abbey to Zoo". In: *CVPR*. 2010.

[116] F Xu and JB Tenenbaum. "Word learning as Bayesian inference". In: *Psychological Review* 114.2 (2007), pp. 245–272.

[117] J Yang, K Yu, and Y Gong. "Linear spatial pyramid matching using sparse coding for image classification". In: *CVPR*. 2009.

[118] J Yang, K Yu, and T Huang. "Efficient highly over-complete sparse coding using a mixture model". In: *ECCV*. 2010.

[119] J Yang et al. "Group-Sensitive Multiple Kernel Learning for Object Categorization". In: *ICCV*. 2009.

[120] J Yang et al. "Linear spatial pyramid matching using sparse coding for image classification". In: *CVPR*. 2009.

[121] K Yu and T Zhang. "Improved local coordinate coding using local tangents". In: *ICML*. 2010.

[122] MD Zeiler, GW Taylor, and R Fergus. "Adaptive Deconvolutional Networks for Mid and High Level Feature Learning". In: *ICCV*. 2011.

[123] K Zhang, IW Tsang, and JT Kwok. "Improved Nyström low-rank approximation and error analysis". In: *ICML*. 2008.

[124] N Zhang et al. "Deformable Part Descriptors for Fine-grained Recognition and Attribute Prediction". In: *ICCV*. 2013.

[125] L Zhu, Y Chen, and A Yuille. "Unsupervised Learning of a Probabilistic Grammar for Object Detection and Parsing". In: *NIPS*. 2007.

[126] H Zou and T Hastie. "Regularization and variable selection via the elastic net". In: *JRSS* 67.2 (2005), pp. 301–320.