THÈSE DE DOCTORAT DE
L'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Hanlin GOH

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ PIERRE ET MARIE CURIE

LEARNING DEEP VISUAL REPRESENTATIONS

APPRENTISSAGE DE REPRÉSENTATIONS VISUELLES PROFONDES

Soutenue à **Paris** le **12 Juillet 2013**, devant le jury composé de:

MM.	Patrick Gallinari,	*Président.*
	Frédéric Jurie,	*Rapporteurs.*
	Alain Rakotomamonjy,	
	Yann LeCun,	*Examinateurs.*
	Joo-Hwee Lim,	
	Matthieu Cord,	*Directeur de thèse.*
	Nicolas Thome,	*Encadrant.*

PARIS

2013

Thèse préparée au sein de les laboratoires suivants:

Laboratoire d'Informatique de Paris 6 (UMR 7606)

Université Pierre et Marie Curie – Sorbonne Universités, Paris, France

Centre National de la Recherche Scientifique, France

4 Place Jussieu, 75005 Paris, France.

Institute for Infocomm Research

Agency for Science, Technology and Research, Singapore

1 Fusionopolis Way, #21-01 Connexis (South Tower), Singapore 138632.

Image and Pervasive Access Laboratory (UMI 2955)

Centre National de la Recherche Scientifique, France

Agency for Science, Technology and Research, Singapore

1 Fusionopolis Way, #21-01 Connexis (South Tower), Singapore 138632.

Partiellement financement géré par l'institut français de Singapour dans le cadre du programme de bourse doctorale «Merlion» (bourse du gouvernement français issue des Partenariat Hubert Curien), de 2010 à 2012.

Institut Français de Singapour

Ambassade de France à Singapour

101-103 Cluny Park Road, Singapore 259595.

Learning Deep Visual Representations ■ Apprentissage de Représentations Visuelles Profondes.

Résumé

Les avancées récentes en apprentissage profond et en traitement d'image présentent l'opportunité d'unifier ces deux champs de recherche complémentaires pour une meilleure résolution du problème de classification d'images dans des catégories sémantiques. L'apprentissage profond apporte au traitement d'image le pouvoir de représentation nécessaire à l'amélioration des performances des méthodes de classification d'images. Cette thèse propose de nouvelles méthodes ***d'apprentissage de représentations visuelles profondes*** pour la résolution de cette tache.

L'apprentissage profond a été abordé sous deux angles. D'abord nous nous sommes intéressés à l'apprentissage non supervisé de représentations latentes ayant certaines propriétés à partir de données en entrée. Il s'agit ici d'intégrer une connaissance à priori, à travers un terme de régularisation, dans l'apprentissage d'une machine de Boltzmann restreinte. Nous proposons plusieurs formes de régularisation qui induisent différentes propriétés telles que la parcimonie, la sélectivité et l'organisation en structure topographique. Le second aspect consiste au passage graduel de l'apprentissage non supervisé à l'apprentissage supervisé de réseaux profonds. Ce but est réalisé par l'introduction sous forme de supervision, d'une information relative à la catégorie sémantique. Deux nouvelles méthodes sont proposées. Le premier est basé sur une régularisation top-down de réseaux de croyance profonds à base de machines des Boltzmann restreintes. Le second optimise un cout intégrant un critère de reconstruction et un critère de supervision pour l'entrainement d'autoencodeurs profonds.

Les méthodes proposées ont été appliquées au problème de classification d'images. Nous avons adopté le modèle sac-de-mots comme modèle de base parce qu'il offre d'importantes possibilités grâce à l'utilisation de descripteurs locaux robustes et de pooling par pyramides spatiales qui prennent en compte l'information spatiale de l'image. L'apprentissage profonds avec agrégation spatiale est utilisé pour apprendre un dictionnaire hiérarchique pour l'encodage de représentations visuelles de niveau intermédiaire. Cette méthode donne des résultats très compétitifs en classification de scènes et d'images. Les dictionnaires visuels appris contiennent diverses informations non-redondantes ayant une structure spatiale cohérente. L'inférence est aussi très rapide. Nous avons par la suite optimisé l'étape de pooling sur la base du codage produit par le dictionnaire hiérarchique précédemment appris en introduisant introduit une nouvelle paramétrisation dérivable de l'opération de pooling qui permet un apprentissage par descente de gradient utilisant l'algorithme de rétro-propagation.

Ceci est la première tentative d'unification de l'apprentissage profond et du modèle de sac de mots. Bien que cette fusion puisse sembler évidente, l'union de plusieurs aspects de l'apprentissage profond de représentations visuelles demeure une tache complexe à bien des égards et requiert encore un effort de recherche important.

Abstract

Recent advancements in the areas of deep learning and visual information processing have presented an opportunity to unite both fields. These complementary fields combine to tackle the problem of classifying images into their semantic categories. Deep learning brings learning and representational capabilities to a visual processing model that is adapted for image classification. This thesis addresses problems that lead to the proposal of ***learning deep visual representations*** for image classification.

The problem of deep learning is tackled on two fronts. The first aspect is the problem of unsupervised learning of latent representations from input data. The main focus is the integration of prior knowledge into the learning of restricted Boltzmann machines (RBM) through regularization. Regularizers are proposed to induce sparsity, selectivity and topographic organization in the coding to improve discrimination and invariance. The second direction introduces the notion of gradually transiting from unsupervised layer-wise learning to supervised deep learning. This is done through the integration of bottom-up information with top-down signals. Two novel implementations supporting this notion are explored. The first method uses top-down regularization to train a deep network of RBMs. The second method combines predictive and reconstructive loss functions to optimize a stack of encoder-decoder networks.

The proposed deep learning techniques are applied to tackle the image classification problem. The bag-of-words model is adopted due to its strengths in image modeling through the use of local image descriptors and spatial pooling schemes. Deep learning with spatial aggregation is used to learn a hierarchical visual dictionary for encoding the image descriptors into mid-level representations. This method achieves leading image classification performances for object and scene images. The learned dictionaries are diverse and non-redundant. The speed of inference is also high. From this, a further optimization is performed for the subsequent pooling step. This is done by introducing a differentiable pooling parameterization and applying the error backpropagation algorithm.

This thesis represents one of the first attempts to synthesize deep learning and the bag-of-words model. This union results in many challenging research problems, leaving much room for further study in this area.

Keywords: Learning deep architectures, deep belief network, restricted Boltzmann machine, sparse coding, selectivity regularization, topographic maps, supervised deep learning, visual dictionary learning, hierarchical visual codes, feature coding, pooling, bag-of-words model, image classification.

Acknowledgements

« La reconnaissance est la memoire du coeur. »
"Gratitude is the memory of the heart"

~ *Jean Baptiste Massieu*

I started this thesis focused on designing new-age connectionist architectures for broad competence vision. After three years of studying the topics, I realize that what I have gained goes beyond the knowledge and skills, but also lifelong friendships for which I am most grateful.

First and foremost, I owe my deepest gratitude to my advisors in Paris, Prof. Matthieu Cord and Dr. Nicolas Thome, and my supervisor in Singapore, Dr. Lim Joo-Hwee. Thank you for the support, guidance and encouragement. I could not have imagined a better combination of advisors, whose knowledge and experiences complement each other so perfectly.

My sincere thanks to the rapporteurs and examiners in my jury, Prof. Frédéric Jurie and Prof. Alain Rakotomanojy, Prof. Yann LeCun and Prof. Patrick Gallinari, for their insightful comments and constructive suggestions.

This thesis would not have started without the financial support by the Embassy of France in Singapore. I am also thankful for the support by the labs I worked in: the Laboratoire d'Informatique de Paris 6 (LIP6), Paris, the Institute for Infocomm Research (I²R), Singapore, and the Image and Pervasive Access Laboratory (IPAL), a French–Singaporean joint laboratory.

Besides my advisors, I thank the researchers and fellow Ph.D. students in the Machine Learning and Information Retrieval (MALIRE) Team in LIP6: Frédéric, Christian, Corina, David, Moustapha, Sandra, Rodrigo, Marc and Jonathan. In particular, Frédéric and Corina helped me settle down in Paris, Christian and Moustapha accommodated my endless translation requests, and Sandra guided me through tedious of the administrative procedures.

I acknowledge my colleagues and students, past and present, in I²R and IPAL. Especially to Isaac, Yilun, Chris, Meng, Tat-Jun, Stanley, Trong Ton, Shijian, Tian-Tsong, Cheston, Jiquan, Weixun, Melvin, Łukasz, Wiktor, Michal, Sepehr, Daniel, Jean-Pierre and Ludovic. A big thank you to all of you for the friendship and endless meetings.

Above all, I thank my family for being the pillars of my life. I thank my parents for guiding me through life and encouraging me to pursuit my dreams. To Jingfen, my wife, thanks for showing patience and giving me moral support.

Paris, July 2013 *Hanlin Goh*

Contents

List of Figures

Page

List of Tables and Algorithms

Page

LEARNING DEEP VISUAL REPRESENTATIONS

APPRENTISSAGE DE REPRÉSENTATIONS VISUELLES PROFONDES

HANLIN GOH

1 | Introduction

Achieving artificial intelligence in computers has been a subject of countless theses, spanning years of research. We have made significant progress in computational "intelligence" via information search techniques, that enable machines to beat chess grandmasters at their game or more recently defeat jeopardy legends. However, searching relies heavily on raw computational power is hardly the solution for many perceptual tasks that are seemingly trivial to us, human beings. This is especially so for computational approaches trying to achieve visual perception and natural language understanding.

This thesis represents the marriage of two fields in artificial intelligence – machine learning and computer vision. While machine learning focuses on making sense of the deluge of available data, computer vision aims to tackle the image understanding challenge, where learning from visual data has been a promising direction. The aspiration is that we can eventually discover a mapping between images and their semantic concepts, either automatically or with as little human intervention as possible. The main objective for this thesis is to investigate how the learning of deep architectures can help improve the performance of computational visual tasks, such as image annotation. The effort is focused on two aspects, namely:

1. Automatic learning of representations for visual information processing, and
2. Construction of hierarchical models to facilitate image annotation.

1.1 The image annotation problem

The last decade has seen the popularization of digital photography. This rise is fueled by the accessibility of image capture using cheap and ubiquitous portable devices, such as digital cameras and mobile phones, as well as the growing ease to share and

view images through the proliferation of social networks and mobile Internet access. Photo sharing is an integral part of the Facebook experience. In 2012, it averaged more than 300 million photo uploads daily, which equates to about 9 billion or 7 petabytes of images per month.[1,2] As of October 2012, it has a total of 220 billion images hosted on its servers. Other photo sharing websites such as Flickr (8 billion) and Instagram (5 billion) also have significantly sized and every expanding image databases.[3,4] With this exploding number of digital images on the Internet, the importance of image annotation becomes ever more significant.

Image annotation is one of the most challenging problems in computer vision and multimedia research. The objective is to map a digital image into one or several labels. This implies understanding complex semantic meaning based on an image's visual content. For example, given the image in Figure 1.1, possible annotations could be objects, such as the "*Eiffel tower*", "*person*", "*tree*" and "*lamp post*", while potential scene labels may be "*park*" and "*outdoor*". Image annotation can be applied to image retrieval, whereby images can be indexed using classes based on their visual content to be used for searching and retrieval at a later time.

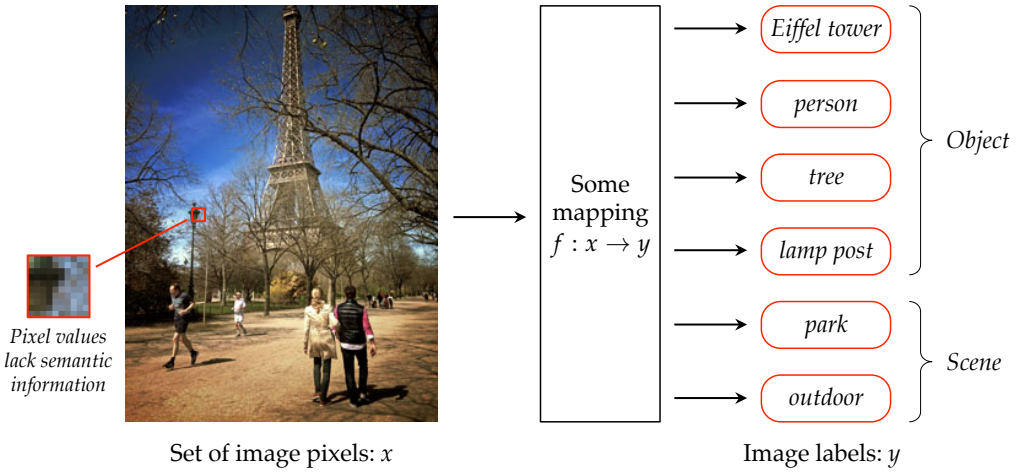


Figure 1.1: The image annotation problem. The challenge of image annotation is to find a mapping that bridges the semantic gap between raw image pixels and semantic concepts, such as objects and scene categories.

The main challenge in image annotation, however, is that raw image pixels do not provide enough unambiguous information to directly generate semantic-level concepts. The challenges are unlike those of text annotation, whereby the dictionary relating words directly to semantics is clearly defined and the syntax is well established to combine alphabets into words and words into sentences. In image annotation, there is no clear-cut definition of 'words' or 'sentences' to associate with the semantics of the

[1]Source: Facebook Developer Blog, July 18, 2012.
[2]Source: GigaOM, October 17, 2012.
[3]Source: Flickr Blog, December 12, 2012.
[4]Source: Instagram Blog, September 6, 2012.

image. This absence of a link between pixels and semantics is known as the 'semantic gap,' as illustrated in Figure 1.1. While the solution to bridge this semantic gap remains elusive, promising research developments in the fields of machine learning and computer vision have been proposed that stride towards this goal.

1.2 History, trends and opportunities

One key focus of this thesis is to learn the mapping between images and their semantics, through a fusion of machine learning and computer vision techniques. Specifically, I explore the combination of recently popularized sparse unsupervised learning and deep learning approaches from machine learning, with the vision-based bag-of-words model for image classification.

Connectionist models: their rise and fall. The current methods for deep learning are an accumulation of much research over the years. Connectionist models popularized in the 1980s, revolutionized the notion of learning distributed representations from data. In particular, fully-connected multilayer perceptron networks, having been shown to be universal approximators, can represent any function with its parameters [Hornik, 1991]. However, the main problems are the huge number of parameters to be learned and the difficult non-convex optimization problem often gets trapped in non-ideal local minima. While the representational power of the model can be theoretically increased with more layers, supervised learning is tedious and difficult to manage for such deep networks [Bengio, 2009].

Moreover, neural networks are often thought to be black boxes that are difficult to understand and train. As such, there was a loss of interest in the neural networks in the 1990s, as more researchers started to favor other statistical learning methods, such as the support vector machine (SVM) [Vapnik, 1995]. SVMs treat the learning problem as a convex optimization and are easy to train. For SVMs to work well, users need to provide complex features or design suitable kernels. However, they also have limited representational power due to its local learning and flat architecture [Bengio, 2009].

Image classification models. In the last decade, various computer vision models were developed by exploiting the then-trendy SVMs for image classification. One particular model, the bag-of-words model, has emerged to achieve good image classification performances on many object and scene image datasets. The model maps from the pixel-level to the semantic-level through a series of data transformation steps, namely: 1) feature extraction, 2) feature coding, 3) pooling and 4) classification.

In a typical setup, gradient-based local image descriptors, such as scale-invariant feature transform (SIFT) [Lowe, 1999] and histogram of orientated gradients (HOG) [Dalal and Triggs, 2005], are used to describe an image. They are discriminative yet robust to various image transformations. A common adaptation for image categorization is the use of spatial pyramids [Lazebnik et al., 2006] to integrate spatial information. In the final step, image classification is generally performed using SVMs.

In the classical formulation, feature coding is generally a fixed (non-learned) and flat (single-layer) operation. The flat structure limits the representational power of model, while the lack of learning makes it difficult to adapt to different data.

Deep learning: a new era in connectionism. While SVMs proliferated within the machine learning and computer vision community, connectionist models were being rejuvenated. The problem of training deep neural networks was recently given a new lease of life, with a focus on unsupervised feature learning. This emphasis on unsupervised learning is crucial because there is usually a lot more unlabeled data compared to labeled ones. The solution to learning multiple layers of representation considers learning each layer of representation as an unsupervised generative module from its input data distribution and stacking them one layer at a time from the bottom-up, in a greedy layer-wise manner [Hinton et al., 2006; Bengio et al., 2006; Hinton, 2007a]. This makes it scale well to deep networks. It also appears sensible to learn simple representations first and higher-level abstractions on top of existing lower level ones. In place of randomly initialized parameters, this unsupervised representation forms the initialization – a catalyst to learn meaningful representations – for the subsequent supervised learning phase.

Deep learning research has led to some success in traditional image classification problems, such as handwritten digit recognition [Hinton et al., 2006]. The increase in interest in this research area has spawned a new conference known as the *International Conference on Learning Representations (ICLR)* in 2013. Deep learning techniques have also won challenges beyond its conventional applicational areas, such as in molecular activity prediction and job salary prediction.[5,6]

In the commercial scene, many technology giants, such as Google, Baidu and Apple, have started actively pursuing research in this area. In 2012, Google acquired DNNResearch, a University of Toronto startup, together with their researchers bringing along their years of experience in the field.[7] Baidu has ambitiously started an

[5] https://www.kaggle.com/c/MerckActivity.
[6] https://www.kaggle.com/c/job-salary-prediction/data
[7] Source: University of Toronto News, March 12, 2013.

Institute for Deep Learning, similarly attracting top researchers worldwide.[8,9]. Deep learning has been implemented to perform speech recognition in Apple's Siri virtual personal assistant, Google Street View and the Android Operating System [Mohamed et al., 2012].[10,11]

Convergence of deep learning and image classification. While the bag-of-words model uses data transformations to map from images to semantics, deep connectionist models learn a mapping from input data to output classes by attempting to untangle the manifold of the highly nonlinear input space. The similarities between the objectives of the two models present opportunities for the fusion of ideas from both methods, by applying connectionist models for image classification.

In the 1980s and 1990s, a classical method known as the convolutional neural network [LeCun et al., 1989, 1998] was developed. It focused on tackling the vision problem through a fully-supervised multilayer network with convolution operators in each layer mapping their inputs to produce a new representation via a bank of filters. Such a highly adapted hierarchical local connectivity has the potential to encode structure suitable for modeling images, such that even with random parameters in the early layer, performance remained impressive [Ranzato et al., 2006]. It produced exceptional performances for specific image datasets [LeCun et al., 1989, 1998].

Recently, the convolutional deep network [Krizhevsky et al., 2012] has emerged as a competitive method for classifying large-scale image datasets with huge amounts of training data [Deng et al., 2009], convincingly winning the *ImageNet Large Scale Visual Recognition Challenge (ILSVRC2012)*, held in conjunction with the *PASCAL Visual Object Classes Workshop (VOC)* at the *European Conference on Computer Vision (ECCV)* in 2012.[12] Due to the representations depth and learning plasticity of these networks, the variety of information it can learn to represent is extensive. However, it requires a lot of labeled training data to avoid model overfitting and substantial computational resources to perform well.

Using unsupervised approaches, the Google Brain project trained a deep neural network on 16,000 CPU cores to automatically learn higher-level visual concepts by watching videos from YouTube. The model was able to successfully learn concepts such as "cats" and "human" [Le et al., 2012]. It also exhibited some success on the large scale ImageNet dataset [Deng et al., 2009].

[8] Source: Sohu IT report, January 21, 2013.
[9] Source: Wired Enterprise, April 12, 2013.
[10] Source: The New York Times, November 23, 2012.
[11] Source: Google Research Blog, August 7, 2012.
[12] `http://www.image-net.org/challenges/LSVRC/2012`

1.3 Contributions

The central contribution of this thesis is the construction and learning of hierarchical architectures for image classification. The contributions can be organized in three parts, namely: 1) learning deep representations, 2) its incorporation to the bag-of-words model and 3) image classification evaluation. I first focus on the extending various aspects of unsupervised feature learning and deep learning. The modified learning algorithms are integrated into the bag-of-words model by learning hierarchical visual dictionaries for feature coding. Additionally, an attempt was made to optimize the pooling step of the bag-of-words model to improve image classification performances. The main contributions of this thesis are listed as follows.

Learning deep representations. The learning of deep representations can be divided into two aspects: unsupervised feature learning and supervised deep learning. I propose a new method to regularize method unsupervised learning that enables better control of the learning process and encourage representations to assume certain desirable coding properties, such as sparsity and selectivity (see Chapter 3). For deep learning, I introduce an original method to integrate supervised top-down information into the learning of representations of the deep network (see Chapter 4). This helps fine-tune the model trained through unsupervised learning.

Image classification. The proposed deep architecture is integrated with the bag-of-words model to classify images from local image descriptors. Both unsupervised feature learning and supervised deep learning, previously explored, are exploited to learn hierarchical visual dictionaries that can be used for the feature coding step (see Chapter 5). In addition, a novel parameterization of the pooling step is suggested together with a method to optimize the pooling parameters (see Chapter 6).

Empirical evaluation. Extensive empirical evaluation was performed on various architecture setups and parameterization. The final proposed model leads to competitive image classification results, outperforming other feature coding methods on both the Caltech-101 [Fei-Fei et al., 2004] and the 15-Scenes [Lazebnik et al., 2006] datasets (see Chapter 5). Competitive results are also achieved on the Caltech-256 [Griffin et al., 2007] dataset. The image representation is compact and non-redundant, as verified both qualitatively and quantitatively. Inference speed is also fast as compared to some existing families of feature coding methods.

1.4 Thesis outline

This work is a multi-disciplinary intersection between machine learning and computer vision. I have structured this thesis in a manner that will hopefully be interesting to a broad range of readers from both the machine learning and computer vision communities.

1.4.1 Recurring themes

The main contribution of this dissertation is the design of deep learning algorithms and architecture to perform image classification, as discussed in this introduction. The following themes fuse the presentation of the design of the methodology and performance analysis in the coming chapters.

- *Modeling and learning.* Both modeling and learning are vital for designing a hierarchical architecture for an image classification task. Modeling entails the capturing of image structure by building virtual linkages to a model. Meanwhile, learning is employed to adapt the parameters of the virtual linkages to extract knowledge or statistics of the image within the structure of the model.

- *Bottom-up and top-down information.* Information for learning visual information and their semantics may come from bottom-up sources, such as image pixels, or from the top-down such as semantic labels. It is critical to consider how this merger of bottom-up and top-down signals can be exploited via a combination of generative and discriminative learning.

- *Biological inspirations.* Since the formation of neural networks, the linkage to neuroscience is apparent. In this thesis, several theoretical and design aspects of the learning algorithms have been inspired by biological phenomenon. However, it is crucial to note that the objective is to improve the learning algorithms rather than build accurate models of neurons and neural systems.

- *Performance evaluation and analysis.* The most tangible demonstration that a method works is to perform experimental evaluation. Empirical studies also provide insight to the effects of parameterization of the model. Depending on the method and task, the performance may be studied based on metrics such as classification accuracy or computational resources, like speed and memory footprint. In some cases, a data visualization may be the most intuitive and effective way to understand the inner workings of a system.

1.4.2 Thesis roadmap

Figure 1.2 is a map of some paths that one may choose to explore the coming chapters. For deep learning researchers, Chapters 3 through to 4 will be a suitable route to take. Meanwhile, Chapters 5 to 6 lead to a more applicational path that is meant for vision practitioners focusing on the bag-of-words model. Finally, those interested in applying the learning deep representations for image classification should find themselves traversing from Chapters 3 and 4 to Chapter 5.

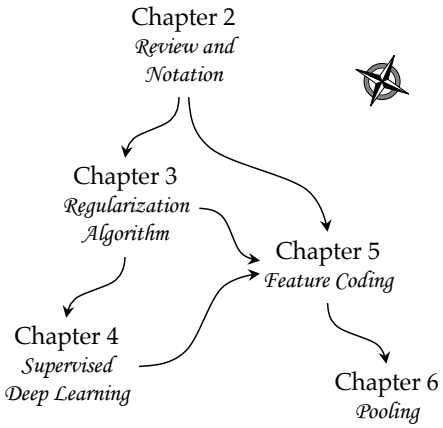


Figure 1.2: Thesis roadmap.

1.4.3 Chapter descriptions

- **Deep Learning and Visual Representations: An Overview.** Chapter 2 presents the motivations and overview of the existing approaches for learning distributed representations and extensions using deep learning strategies. The chapter also introduces the bag-of-words model, which is a hierarchical model that transforms visual representations for image classification.

- **Regularizing Latent Representations.** Chapter 3 proposes a regularization method for unsupervised learning to form representations that assume desirable properties for various vision tasks. The biologically-inspired coding properties such as sparsity, selectivity and topographic organization are explored in detail.

- **Deep Supervised Optimization.** Chapter 4 introduces a deep learning strategy that gradually transits from unsupervised to supervised learning. To achieve this, the regularization method of Chapter 3 is extended to integrate bottom-up information with top-down signals. The overall deep learning strategy is also investigated with a deep encoder-decoder network.

- **Learning Hierarchical Visual Codes.** Chapter 5 applies the unsupervised learning method of Chapter 3 to the bag-of-words model for the purposes of dictionary learning and feature coding. The subsequent development into a hierarchical feature coding architecture is also explored by using a supervised learning method in Chapter 4.

- **Discriminative Pooling.** Chapter 6 continues the work of Chapter 5, this time focusing on optimizing the pooling step of the bag-of-words model.

- **Conclusions.** Chapter 7 concludes this thesis and suggests future directions for exploration. This thesis has led to the publication of one journal article and four international peer-reviewed conference papers. A complete list of the publications is also included in this chapter.

2 Deep Learning and Visual Representations

— An Overview

Chapter abstract — This thesis marries deep learning with modeling visual representations for image classification. In this chapter, I will provide an overview of the motivations and methods for each research area. First, I present a historical perspective of connectionist models that learn distributed representations from data. The focus will be on recent unsupervised learning methods that discover structure in the input data and learn latent representations. These unsupervised methods serve as building blocks for learning deep architectures. The motivations and strategies for deep learning will also be discussed. I will then present the application of some of these hierarchical architectures to model images for image classification, such as convolutional networks. Subsequently, the popular vision-based bag-of-words model, which uses a multi-step approach to perform data transformation of visual representations from image pixels to semantic categories for image classification, is described. It will be interesting to see how these models focusing on various aspects of learning and vision can complement each other and ultimately lead to hybrid models for image classification.

2.1 Introduction

Machine learning and computer vision research have been advancing hand-in-hand for the last decade. Machine learning focuses on understanding the theory of learning and developing algorithms that can extrapolate from data, while computer vision attempts to solve challenging and practical problems relating to visual perception. Recent developments, specifically in the aspects of deep learning and modeling visual representations, present an opportunity to unite the strengths of both approaches to tackle the image annotation problem (Section 1.1).

Deep learning focuses on learning multiple layers of distributed representations that untangle the manifold from input data to corresponding data classes. Meanwhile, one of the approaches for image classification is to model images with visual representations that map from image pixels to visual semantics. The similarity in objectives and general approach to the classification problem motivates the hybridization of techniques from both approaches.

2.2 Learning distributed representations

A straightforward scheme to represent entities is to use one computational unit to represent each entity. This is known as a local representation. A simplistic way to present a local coding is using a one-hot coded vector of J bits, with each bit representing an input pattern. This is sometimes also known as the grandmother cell [Konorski, 1967], where the representation essentially memorizes input templates. This scheme does not scale well when the number of input patterns increases.

Rather than using a single computational element, connectionist models represent each conceptual entity as a distributed pattern of activity, known as distributed representation [Hinton et al., 1986]. A distributed representation, is exponentially more compact. Moreover, such a coding scheme is able to handle the curse of dimensionality and have the nice property of graceful degradation. For the rest of this section, I will introduce various supervised and unsupervised connectionist models to learn distributed representations.

2.2.1 Feedforward neural networks

A feedforward neural network is one of the most basic types of artificial neural networks, designed to learn distributed representations of input data. The network consists of a series of information processing operations, whereby information moves strictly in the forward direction, from the input units, through latent units, if any, and finally to output units.

Perceptron network. The simplest feedforward neural network is the perceptron network [Rosenblatt, 1958], which builds upon a mathematical model of a neuron of McCulloch and Pitts [1943]. As illustrated in Figure 2.1, output value $\hat{y}$ of the input vector $\mathbf{x}$ is generated with a set of weight parameters $\mathbf{w}$ and a feedforward activation function f_e. Here, w_0 is a special parameter known as the offset or bias, whereby it connects a unit x_0 that is permanently set to 1. For an input with I dimensions, the outputs may be linearly activated through a weighted sum of inputs

$$\hat{y} = f_e(\mathbf{x}, \mathbf{w}) = \sum_{i=0}^{I} w_i x_i, \tag{2.1}$$

or binarized by a hard limiter:

$$\hat{y} = f_e(\mathbf{x}, \mathbf{w}) = \begin{cases} 1 & \text{if } \sum_{i=0}^{I} w_i x_i > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

A continuous approximation of this step function bounds the output through a non-linear squashing function using either the logistic (or sigmoid) function:

$$\hat{y} = f_e(\mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp(-\sum_{i=0}^{I} w_i x_i)}, \tag{2.3}$$

or the hyperbolic tangent function:

$$\hat{y} = f_e(\mathbf{x}, \mathbf{w}) = \tanh\left(\sum_{i=0}^{I} w_i x_i\right). \tag{2.4}$$

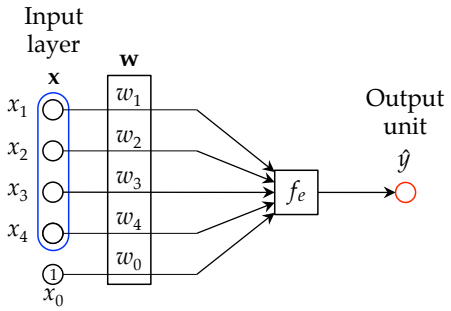


FIGURE 2.1: The perceptron network. The network is based on the model of a neuron by McCulloch and Pitts [1943], uses a set of weights $\mathbf{w}$ and a feedforward activation function f_e to map a layer of input units $\mathbf{x}$ to the output unit $\hat{y}$.

The delta learning rule is used to train a perceptron with continuous differentiable activation functions. The weights must first be initialized to 0 or a small random value around 0. Given a set of training data $\mathcal{D}_{train}$ of input-output pairings $\{(\mathbf{x}_k, y_k)\}$, the weights can be updated for each example k:

$$w_i := w_i + \varepsilon(y_k - \hat{y}_k)x_{ik}, \tag{2.5}$$

where ε is a learning rate. The process can be iterated until the total error of the entire dataset falls below a certain threshold. If the data is linearly separable, the perceptron learning algorithm converges to the solution after a finite number of iterations [Novikoff, 1962]. If the sigmoid function is used, then this gradient descent update (Equation 2.5) is governed by the cross-entropy loss function

$$\begin{aligned} \mathcal{L}_{net} &= -\sum_{k=1}^{|\mathcal{D}_{train}|} \log P(y_k|\hat{y}_k) \\ &= -\sum_{k=1}^{|\mathcal{D}_{train}|} y_k \log \hat{y}_k + (1 - y_k)\log(1 - \hat{y}_k). \end{aligned} \tag{2.6}$$

A multiclass perceptron is a natural extension to handle multiclass datasets $\{(\mathbf{x}_k, \mathbf{y}_k)\}$. The output is a one-hot coded vector $\mathbf{y}_k$ of C units with each representing a class, such

that $y_{ck} = 1$ if c is the class of the training example k, and is otherwise 0. The predicted class for k is given by $\arg\max_c \hat{y}_{ck}$. The weight parameters are structured as a matrix $\mathbf{W} \in \mathbb{R}^{(I+1)\times C}$ rather than a vector. Additionally, the softmax activation function

$$\hat{y}_c = f_{e,c}(\mathbf{x}, \mathbf{W}) = \frac{\exp(\sum_{i=0}^{I} w_{ic} x_i)}{\sum_{\hat{c}=1}^{C} \exp(\sum_{i=0}^{I} w_{i\hat{c}} x_i)} \tag{2.7}$$

is the generalization of the sigmoid function for multiple classes. The function normalizes the output layer such that the units are between 0 and 1 and they sum to 1. As a result, the softmax output $\hat{y}_c$ can be considered as an estimator of $P(\hat{y} = c|\mathbf{x})$.

The same cross-entropy loss function is ideal for updating the weight parameters:

$$\mathcal{L}_{net} = -\sum_{k=1}^{|\mathcal{D}_{train}|} \sum_{c=1}^{C} \log P(y_{ck}|\hat{y}_{ck}). \tag{2.8}$$

resulting in the following weight update rule:

$$\begin{aligned} w_{ic} &:= w_{ic} + \varepsilon \frac{\partial \mathcal{L}_{net}}{\partial w_{ic}} \\ &:= w_{ic} + \varepsilon (y_{ck} - \hat{y}_{ck}) x_{ik}, \end{aligned} \tag{2.9}$$

which has the same form as the perceptron learning rule for a single output unit.

Multilayer perceptron network. The perceptron network can be stacked into a multilayer network known as the multilayer perceptron (MLP), whereby units in successive layers are fully connected to each other. Figure 2.2 shows a three-layer MLP, which has an intermediate latent layer connecting the input and output layers.

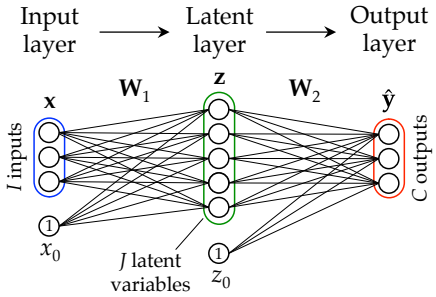


Figure 2.2: Multilayer perceptron. A three-layer multilayer perceptron uses two successive steps of weighed data transformation to map from inputs to the outputs.

If the activation functions of units in the network are differentiable, then the network is trained via gradient descent using the error backpropagation algorithm [LeCun, 1985; Rumelhart et al., 1986]. Consider a three-layer network (Figure 2.2) with the input layer $\mathbf{x} \in \mathbb{R}^I$ and latent layer $\mathbf{z} \in \mathbb{R}^J$ linked by weights $\mathbf{W}_1$, and the latent layer linked to the output layer $\hat{\mathbf{y}} \in \mathbb{R}^C$ via weights $\mathbf{W}_2$. Given a loss function $\mathcal{L}_{net}$, such as

the cross-entropy loss or the squared loss, the weights of the network can be updated by computing the gradients as follows:

$$w_{2,jc} := w_{2,jc} + \varepsilon \frac{\partial \mathcal{L}_{net}}{\partial w_{2,jc}},$$
$$w_{1,ij} := w_{1,ij} + \varepsilon \frac{\partial \mathcal{L}_{net}}{\partial z_j} \frac{\partial z_j}{\partial w_{1,ij}}. \tag{2.10}$$

The chain rule is exploited to compute the gradient of the first set of weights $\mathbf{W}_1$, while $\mathbf{W}_2$ is updated in the same manner as the single-layer perceptron.

Unlike the perceptron network, the multilayer perceptron is able to learn a model of data that is not linearly separable. A three-layer MLP is considered to be a universal approximator of continuous functions. This was proven by Cybenko [1989] for sigmoid activation function and by Hornik [1991] for the multilayer feedforward network in general, regardless of the choice of activation function.

When the size of the MLP grows beyond three representational layers, the optimization problem becomes harder. Bengio and LeCun [2007] suggested that this is due to the training getting trapped in a local minima or plateaus that produce worse results as compared to shallower networks. As illustrated in Figure 2.3, when more layers are added into the network, the error signal become diluted at the lower layers. The issue of a large fan-in from the intermediate units, results in an averaging effect of the error-based gradients that is magnified through multiple layers. Section 2.3 discusses methods to circumvent these limitations of architectural depth.

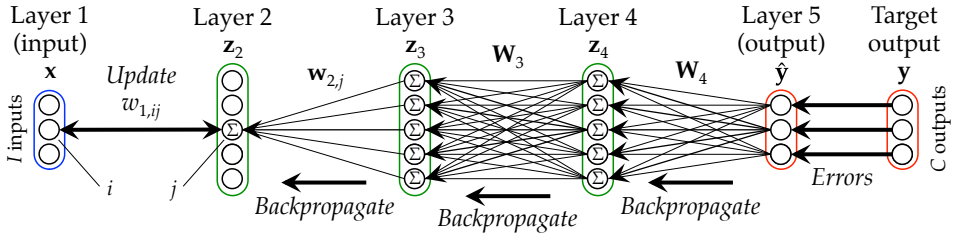


FIGURE 2.3: Diluted error signal of a deep network. With the error backpropagation algorithm, the lower layers receive error signals that are diluted through repeated signal averaging by multiple layers of units with high fan-in.

2.2.2 Auto-associative networks

This section focuses on three types of auto-associative neural networks, namely 1) the auto-encoder network, 2) the decoder network, and 3) the encoder-decoder network. These three networks focus on learning distributed representations that can reconstruct input examples.

Auto-encoder network. An auto-encoder network [Bourlard and Kamp, 1988] is a variant of a three-layer MLP, which models a set of inputs through unsupervised learning. The auto-encoder learns a latent represent that performs reconstruction of the input vectors (see Figure 2.4). This means that the input and output layers assume the same conceptual meaning. Given an input $\mathbf{x}$, a set of weights $\mathbf{W}$ maps the inputs to the latent layer $\mathbf{z}$. From this latent layer, the inputs $\hat{\mathbf{x}}$ are reconstructed with another set of weights $\mathbf{V}$. To train the auto-encoder, the target vectors are the inputs themselves. Bourlard and Kamp [1988] showed that for an auto-encoder with linearly activated units and using the squared loss error function, then the optimal solution for $\mathbf{W}$ and $\mathbf{V}$ can be obtained by the singular value decomposition (SVD) corresponding to the principle component analysis (PCA), without requiring to perform gradient descent.

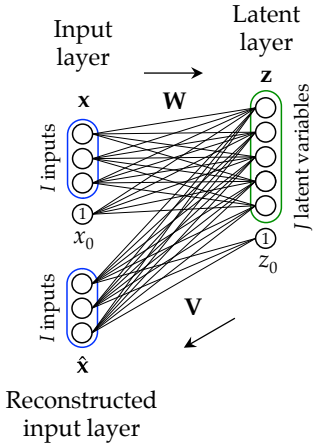


Figure 2.4: Autoencoder network. The network maps back to the inputs through a latent layer and uses input reconstruction errors to update its parameters.

Originally, auto-encoders have a latent layer smaller than the dimensionality of the input layer to perform data compression. Meanwhile, Bengio and LeCun [2007] suggested using a latent layer larger than the input layer to learn representations for feature coding. However, an auto-encoder with the number of latent units greater than the number of input dimensions (i.e. $J > I$) can potentially learn trivial solutions such as the identity function. As such, the auto-encoder can be constrained or regularized during learning to prevent such scenarios [Collobert and Bengio, 2004]. Another effective way to prevent learning the identity function is to use the denosing auto-encoder [Vincent et al., 2008], which adds noise to the encoding $\mathbf{x}$ but tries to reconstruct the clean input through the latent representation.

Decoder network. Just like auto-encoder networks, decoder networks perform learning from unlabeled data through unsupervised learning based on input reconstruction. However, the learning algorithm is approached from a reverse direction. There are two representational layers in the decoder network, linked by a set of weights. Suppose the inputs $\mathbf{x}$ can be represented as a linear combination of basis functions

each given by $\mathbf{w}_i \in \mathbb{R}^J$ through a decoding function:

$$\hat{x}_i = f_d(\mathbf{z}, \mathbf{W}) = \sum_{j=1}^{J} w_{ij} z_j, \tag{2.11}$$

where $\hat{x}_i$ is a reconstructed input and z_j can be seen as coding coefficients for the basis function w_{ij}.

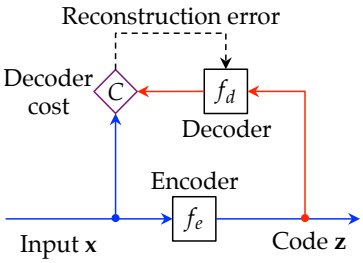


FIGURE 2.5: Decoder network. The network uses the reconstruction error to learn both the parameters and latent code.

The loss function of the network can be defined as the reconstruction error (see Figure 2.5) using the mean square error:

$$\mathcal{L}_{net} = \|\mathbf{x}_k - f_d(\mathbf{z}_k, \mathbf{W})\|_2^2. \tag{2.12}$$

The optimal coding coefficients for reconstructing the inputs can be obtained through gradient descent

$$\begin{aligned} z_{jk} &:= z_{jk} + \eta \frac{\partial \mathcal{L}_{net}}{\partial z_j} \\ &:= z_{jk} + \eta (x_{ik} - \hat{x}_{ik}) w_{ij}, \end{aligned} \tag{2.13}$$

where η defines the size of each gradient descent step for updating the coding. Meanwhile, since the weights are not known, an optimization is also required to learn them. This can again be done by gradient descent, based on the partial derivative $\frac{\partial \mathcal{L}_{net}}{\partial w_{ij}}$.

$$\begin{aligned} w_{ij} &:= w_{ij} + \varepsilon \frac{\partial \mathcal{L}_{net}}{\partial w_{ij}} \\ &:= w_{ij} + \varepsilon (x_{ik} - \hat{x}_{ik}) z^*_{jk}, \end{aligned} \tag{2.14}$$

where z^*_{jk} is an optimized coding from Equation 2.13. The result of these two updates is an algorithm, that performs optimization on both the coding and the weights. In computer vision, this set of weights is also known as a dictionary or codebook. Algorithm 2.1 shows one example of this algorithm that alternates between code optimization and dictionary learning. Recently, Rakotomamonjy [2013] showed that it might

not be necessary to perform alternating optimization between the two sets of parameters or even a full optimization for each set. Rather, the dictionary can be learned through direct optimization of the non-convex function using a block-coordinate proximal gradient descent method.

Algorithm 2.1: Decoder network optimization

```
Initialize W
repeat
    Get training input x_k
    Fix W
    repeat                                          // Code optimization
        Computer ||x_k − f_d(z_k, W)||^2
        Update z_k with 1-step gradient descent     // Equation 2.13
    until stopping criteria is met
    Fix optimal codes z*_k
    repeat                                          // Dictionary learning
        Computer ||x_k − f_d(z*_k, W)||^2
        Update W with 1-step gradient descent       // Equation 2.14
    until stopping criteria is met
until convergence
```

Sparse regularization. Similar to auto-encoders, decoder networks can also be regularized to generate more meaningful representations. A popular regularizer is sparsity, first introduced by Olshausen and Field [1996]. The resulting optimization couples the information preservation term (Equation 2.13)

$$\mathcal{L}_{net} = \|\mathbf{x}_k - f_d(\mathbf{z}_k, \mathbf{W})\|_2^2 + \lambda h(\mathbf{z}_k), \tag{2.15}$$

where λ is a regularization constant. Olshausen and Field [1996] suggested the use of various types of regularization terms such as $-\sum_{j=1}^{J} \exp(\frac{z_j}{\psi})$, $\sum_{j=1}^{J} \log(1 + (\frac{z_j}{\psi})^2)$ and $\sum_{j=1}^{J} |\frac{z_j}{\psi}|$ where ψ is a scaling constant. The $\sum_{j=1}^{J} \log(1 + (\frac{z_j}{\psi})^2)$ regularizer corresponds to a Student's t-distributed prior on z_j. This optimization looks for a minimum-entropy code [Barlow, 1989], based on a conjuncture that natural images have a naturally sparse structure. Interestingly, the resulting dictionary learned correspond to the receptive field encodings of simple cells in the V1 region of our visual cortex [Olshausen and Field, 1996]. Sparse dictionary learning using decoder networks has also gained popularity for encoding image features for image classification (see Section 2.4.3).

A direct measure of sparsity is the ℓ_0-norm of the vector $\mathbf{z}_k$, which counts number of nonzero elements. The sparsest representation is given by the minimum ℓ_0-norm. However, the optimization is highly non-convex and often intractable. Instead, surrogate functions, such as the ℓ_1-norm, are used to approximate the ℓ_0-norm. The

ℓ_1-norm regularized optimization of the decoder network for generating sparse codes can be written as follows:

$$\mathbf{z}_k^* = \|\mathbf{x}_k - f_d(\mathbf{z}_k, \mathbf{W})\|_2^2 + \lambda\|\mathbf{z}_k\|_1. \tag{2.16}$$

This setup has been effective in producing sparse codes for modeling images [Mairal, 2010; Boureau et al., 2010a; Kavukcuoglu et al., 2010]. The main algorithms used to solve the optimization problem include the coordinate gradient descent based on the least absolute shrinkage and selection operator (LASSO) formulation by Fu [1998], the least angle regression (LARS) method of Efron et al. [2004], the fast proximal gradient method known as fast iterative shrinkage-thresholding algorithm (FISTA) by Beck and Teboulle [2009] and the fast approximate structured sparse coding by Szlam et al. [2012].

Encoder-decoder networks. Although the decoder network is able to learn meaningful representations, especially with sparse regularization, the inference process is slow because the sparse coding optimization needs to be solved separately for every input example k, as evident in Equation 2.16. An alternative to improve inference speed is to introduce a variant of the decoder network, known as the encoder-decoder network [Ranzato et al., 2006]. The encoder-decoder network [Ranzato et al., 2006] is an extension of the decoder network [Olshausen, 2001] that concurrently learns the transformation of the input vector to the representation through an encoder.

The input layer $\mathbf{x}$ and latent layer $\mathbf{z}$ are linked via forward weights $\mathbf{W} \in \mathbb{R}^{I+1\times J+1}$ and backward connections $\mathbf{V} \in \mathbb{R}^{I+1\times J+1}$, where v_{i0} and w_{0j} are input and output biases respectively, connected to x_0 and y_0, which are always set to one. It is also possible design choice for the same encoder-decoder network to have shared weights such that $w_{ij} = v_{ij}$. For the purposes of generality, separate weights will be used for this discussion.

The latent layer is activated from the input layer using an encoding function

$$\mathbf{z} = f_e(\mathbf{W}, \mathbf{x}) = \mathbf{W}^T\mathbf{x}, \tag{2.17}$$

for a linear encoder. Similarly, the input layer is activated from the latent layer by a linear decoder with the decoding function

$$\hat{\mathbf{x}} = f_d(\mathbf{V}, \mathbf{z}) = \mathbf{V}\mathbf{z}. \tag{2.18}$$

The system (Figure 2.6) is governed by a loss function $\mathcal{L}_{net}$ that is low when the pair of input and latent vectors $(\mathbf{x}, \mathbf{z})$ exhibit compatibility or likelihood with respect to the

set of parameters $\{\mathbf{W}, \mathbf{V}\}$. The loss function can be defined as a linear combination of costs from the encoder $\mathcal{L}_{enc}$ and decoder $\mathcal{L}_{dec}$:

$$\begin{aligned}\mathcal{L}_{net} &= \lambda_e \mathcal{L}_{enc}(\mathbf{x}, \mathbf{W}) + \lambda_d \mathcal{L}_{dec}(\mathbf{z}, \mathbf{V}) \\ &= \lambda_e \|\mathbf{z} - f_e(\mathbf{x}, \mathbf{W})\|_2^2 + \lambda_d \|\mathbf{x} - f_d(\mathbf{z}, \mathbf{V})\|_2^2,\end{aligned} \tag{2.19}$$

where λ_e and λ_d are parameters proportional to the respective learning rates. The first term, known as the code prediction cost, attempts to make the code $\mathbf{z}$ similar to the output of the encoder. The second term is the reconstruction cost, which tries to minimize the reconstruction error of $\mathbf{x}$. Thus, the optimization concurrently learns both the encoder and the decoder by minimizing the loss function. Additionally, to learn sparse codes, Ranzato et al. [2006] applied a transform of the code $\mathbf{z}$ prior to the decoding operation.

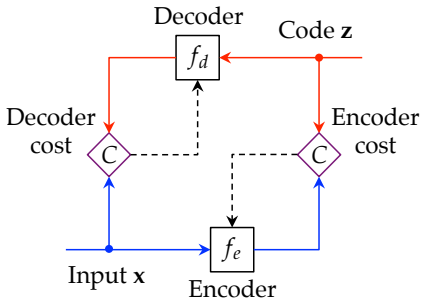


Figure 2.6: The encoder-decoder network. The result of encoding an input $\mathbf{x}$ is compared against the code $\mathbf{z}$ to produce the updates for the encoder. The difference between input $\mathbf{x}$ and the reconstructions by the decoder is used to update the decoding parameters.

If the set of parameters $\{\mathbf{W}, \mathbf{V}\}$ are fixed, it is possible to optimize for the codes $\mathbf{z}$. Similarly, if the coding $\mathbf{z}$ is known, then it is easy to minimize with respect to $\mathbf{W}$ and $\mathbf{V}$. To optimize the network, one can employ coordinate gradient descent algorithm, whereby an inner loop in the iterative process alternates between optimizing for $\mathbf{z}$ with fixed $\{\mathbf{W}, \mathbf{V}\}$, and updating $\{\mathbf{W}, \mathbf{V}\}$ based on the optimized $\mathbf{z}^*$. Because the encoding weights $\mathbf{V}$ are learned, inference is a simple and fast feedforward operation, rather than a complex sparse re-optimization in the case of the decoder networks.

2.2.3 Boltzmann distribution density models

Besides auto-encoders, decoder networks and encoder-decoder networks (Section 2.2.2), another commonly used neural network for unsupervised feature learning is the restricted Boltzmann machine (RBM) [Smolensky, 1986]. An RBM is a variant of the Boltzmann machine [Hinton and Sejnowski, 1986]. These two probability density models fall into the family of recurrent neural networks with dynamics governed by

Lyapunov (or energy) functions. Specifically, the probability density is given by the Boltzmann (or Gibbs) distribution.

Boltzmann machines. The Boltzmann machine [Hinton and Sejnowski, 1986] contains I stochastic binary units x_i that are fully connected to each other, as illustrated in Figure 2.7(a). It has the same fully-connected structure as the Hopfield [1982] network, but the activation and learning rules are different. The probability of activating a unit x_i in the Boltzmann machine is given by the sigmoid function:

$$P(x_i = 1; \mathbf{W}) = \frac{1}{1 + \exp(-\sum_{j=0}^{I} w_{ij} x_j)}. \tag{2.20}$$

The network models the probability distribution of input vectors $\mathbf{x}$ based on the Boltzmann (or Gibbs) distribution given by

$$P(\mathbf{x}) = \frac{\exp(-E(\mathbf{x}))}{\sum_{\mathbf{x}} \exp(-E(\mathbf{x}))}, \tag{2.21}$$

where the energy function $E(\mathbf{x})$ is described by

$$E(\mathbf{x}) = -\sum_{i<j} x_i w_{ij} x_j \tag{2.22}$$

Given a set of training inputs $\mathcal{D}_{train} = \{\mathbf{x}_k : k \in [1, |\mathcal{D}_{train}|]\}$, the learning objective is to find the set of weight parameters that the input vectors $\mathbf{x}_k$ have a high average probability under the Boltzmann distribution. For computational speed and accuracy, it is useful to express the optimization in terms of the data log-likelihood:

$$\begin{aligned} -\log P(\mathbf{x}) &= -\log \frac{\exp(-E(\mathbf{x}))}{\sum_{\mathbf{x}} \exp(-E(\mathbf{x}))} \\ &= -\log \exp(-E(\mathbf{x})) + \log \sum_{\mathbf{x}} \exp(-E(\mathbf{x})) \end{aligned} \tag{2.23}$$

By taking the partial derivative with respect to w_{ij}, we get

$$\begin{aligned} -\left\langle \frac{\partial \log P(\mathbf{x})}{\partial w_{ij}} \right\rangle_{data} &= -\left\langle \frac{\partial \log \exp(-E(\mathbf{x}))}{\partial w_{ij}} \right\rangle_{data} + \left\langle \frac{\partial \log \sum_{\mathbf{x}} \exp(-E(\mathbf{x}))}{\partial w_{ij}} \right\rangle_{model} \\ &= \left\langle \frac{\partial E(\mathbf{x})}{\partial w_{ij}} \right\rangle_{data} - \left\langle \frac{\partial E(\mathbf{x})}{\partial w_{ij}} \right\rangle_{model} \\ &= \langle x_i x_j \rangle_{data} - \langle x_i x_j \rangle_{model}, \end{aligned} \tag{2.24}$$

where $\langle \cdot \rangle_{data}$ is the expectation of the input data distribution, while $\langle \cdot \rangle_{model}$ is the expectation of the stationary distribution of the model. The weights of the network

can be updated using gradient descent

$$w_{ij} := w_{ij} + \varepsilon \left(\langle x_i x_j \rangle_{data} - \langle x_i x_j \rangle_{model} \right). \tag{2.25}$$

In general, the first term $\langle x_i x_j \rangle_{data}$ can be computed easily by sampling from the data distribution. However, there is no efficient way to computer the later term $\langle x_i x_j \rangle_{model}$. To produce an estimate of the average over the sampled distribution $P(\mathbf{x})$, a Markov Chain Monte Carlo approach can be used to run the network to the equilibrium distribution. However, this is very slow because it requires prolonged Gibbs sampling to explore the distribution $P(\mathbf{x})$ and arrive at the equilibrium. Furthermore, it is challenging to determine if the equilibrium distribution has indeed been reached.

Restricted Boltzman machines. The training of a Boltzmann machine requires obtaining samples from distributions in the network, but this is also the hardest to obtain due to its fully connected structure (Figure 2.7(a)). To ease the sampling problem yet still be able to model interesting distributions, a variant known as the restricted Boltzmann machine (RBM) [Smolensky, 1986] is used. The RBM is a bipartite Markov random field with a layer of visible input units and a layer of latent units (Figure 2.7(b)). It is fully connected between layers, but have no intra-layer connections. Figure 2.7 compares the structures of the Boltzmann machine and the RBM.

(a) Network of fully-connected units. (b) Bipartite structure of the restricted Boltzmann machine.

Figure 2.7: Structural differences between the bipartite restricted Boltzmann machines and fully-connected networks, such Boltzmann machines and Hopfield networks. An RBM consists of two subsets (or layers) of units that are not connected within each layer, but are fully connected with units in the other layer.

The result is a latent layer that models interactions between input dimensions. This form of modeling leads to a variety of applications, such as dimensionality reduction [Hinton and Salakhutdinov, 2006], classification [Larochelle and Bengio, 2008], collaborative filtering [Salakhutdinov et al., 2007], hashing [Salakhutdinov and Hinton, 2009] and, most importantly for our context, unsupervised feature learning [Hinton et al., 2006; Lee et al., 2008]. It also forms the building block for a deep architecture known as the deep belief network [Hinton et al., 2006] (see Section 2.3.2).

Consider a binary RBM with a visible input layer $\mathbf{x}$ and a latent layer $\mathbf{z}$, as shown in Figure 2.8. The input layer contains I dimensions corresponding to the size of the

input vector. The latent layer has J latent variables. Additionally, there are offset (or bias) units, x_0 and z_0, that are permanently set to one. The layers are associated by an undirected weight matrix $\mathbf{W}$, such that every input unit i is connected to every latent variable j via w_{ij}.

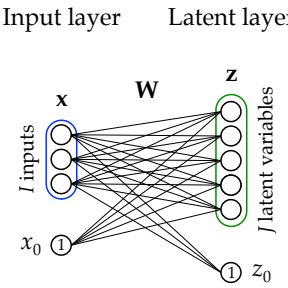


FIGURE 2.8: Structure of the restricted Boltzmann machine (RBM). The RBM connects an input layer $\mathbf{x}$ to a latent layer $\mathbf{z}$ via undirected weights $\mathbf{W}$ and biases x_0 and z_0.

The bipartite structure with no intra-layer connections means that the units in the latent layer $\mathbf{z}$ are independent, given the input layer $\mathbf{x}$. The units in the input layer $\mathbf{x}$ are also conditionally independent, given $\mathbf{z}$. This simplifies the Gibbs sampling process. Given an input vector, the activation probabilities of the latent units can be sampled:

$$P(z_j = 1 \mid \mathbf{x}; \mathbf{W}) = \frac{1}{1 + \exp(-\sum_{i=0}^{I} w_{ij} x_i)}. \tag{2.26}$$

While the input units can be sampled from the latent vector with a symmetric decoder:

$$P(x_i = 1 \mid \mathbf{z}; \mathbf{W}) = \frac{1}{1 + \exp(-\sum_{j=0}^{J} w_{ij} z_j)}. \tag{2.27}$$

For a binary RBM, the joint configuration $(\mathbf{x}, \mathbf{z})$ of activation states in the network has an energy given by:

$$E(\mathbf{x}, \mathbf{z}) = -\sum_{i=0}^{I} \sum_{j=0}^{J} x_i w_{ij} z_j. \tag{2.28}$$

Unlike the Boltzmann machine, the RBM models the joint probability distribution of states corresponding to the energy function $E(\mathbf{x}, \mathbf{z})$ as follows:

$$P(\mathbf{x}, \mathbf{z}) = \frac{\exp\left(-E(\mathbf{x}, \mathbf{z})\right)}{\sum_{\mathbf{x},\mathbf{z}} \exp\left(-E(\mathbf{x}, \mathbf{z})\right)}, \tag{2.29}$$

where the denominator $\sum_{\mathbf{x},\mathbf{z}} \exp\left(-E(\mathbf{x}, \mathbf{z})\right)$ is a normalization constant known as the partition function. Based on the maximum likelihood principle, the objective of the network is to maximize the marginal probability of the input data $\mathbf{x}$ by summing over all possible vectors in the latent layer $\mathbf{z}$:

$$P(\mathbf{x}) = \frac{\sum_{\mathbf{z}} \exp\left(-E(\mathbf{x}, \mathbf{z})\right)}{\sum_{\mathbf{x},\mathbf{z}} \exp\left(-E(\mathbf{x}, \mathbf{z})\right)}. \tag{2.30}$$

Given a set of training set $\mathcal{D}_{train}$ of input vectors $\{\mathbf{x}_k : k \in [1, |\mathcal{D}_{train}|]\}$, the objective is to find the set of weight parameters that can minimize the average negative log-likelihood of the input data:

$$\begin{aligned}\mathbf{W}^* &= \arg\min_{\mathbf{W}} -\langle \log P(\mathbf{x}) \rangle_{data} \\ &= \arg\min_{\mathbf{W}} -\left\langle \log \sum_{\mathbf{z}} \exp\left(-E(\mathbf{x}, \mathbf{z})\right) \right\rangle_{data} + \left\langle \log \sum_{\mathbf{x},\mathbf{z}} \exp\left(-E(\mathbf{x}, \mathbf{z})\right) \right\rangle_{model} \end{aligned} \tag{2.31}$$

The partial derivative of the objective function with respect to w_{ij} is

$$\begin{aligned} -\left\langle \frac{\partial \log P(\mathbf{x})}{\partial w_{ij}} \right\rangle_{data} &= -\left\langle \frac{\partial \log \sum_{\mathbf{z}} \exp\left(-E(\mathbf{x}, \mathbf{z})\right)}{\partial w_{ij}} \right\rangle_{data} + \left\langle \frac{\partial \log \sum_{\mathbf{x},\mathbf{z}} \exp\left(-E(\mathbf{x}, \mathbf{z})\right)}{\partial w_{ij}} \right\rangle_{model} \\ &= \left\langle \frac{\partial E(\mathbf{x}, \mathbf{z})}{\partial w_{ij}} \right\rangle_{data} - \left\langle \frac{\partial E(\mathbf{x}, \mathbf{z})}{\partial w_{ij}} \right\rangle_{model} \\ &= \langle x_i z_j \rangle_{data} - \langle x_i z_j \rangle_{model}, \end{aligned} \tag{2.32}$$

where $\langle \cdot \rangle_{dist}$ denotes the expectation under the distribution $dist$. The first term increases the probability of data driven activations that are clamped by the environment, while the second term reduces the probability of model driven states are sampled from the equilibrium distribution of a free running network.

Contrastive divergence learning. Minimizing the average negative log-likelihood of the data distribution is the same as minimizing the Kullback-Leibler divergence [Kullback and Leibler, 1951] between the data distribution $P_0(\mathbf{x})$ (at time $t = 0$) and the equilibrium distribution $P_\infty(\mathbf{x})$ (at time $t = \infty$):

$$D_{KL}(P_0 \| P_\infty) = \sum_{(\mathbf{x},\mathbf{y})} P_0(\mathbf{x}, \mathbf{y}) \log \frac{P_0(\mathbf{x}, \mathbf{y})}{P_\infty(\mathbf{x}, \mathbf{y})} \geq 0. \tag{2.33}$$

Hinton [2002] proposed the contrastive divergence learning algorithm, that approximates the equilibrium distribution with a small finite number of sampling steps. The Markov chain is relaxed to run for N of sampling steps to generate a reconstruction of data vectors.[1] This sampling approximation is illustrated in Figure 2.9. Perpiñán and Hinton [2005] showed that even by using a single Gibbs sampling step ($N = 1$), the algorithm generates only a small bias [Bengio and Delalleau, 2009]. The optimization seeks to minimize the difference between the two divergences:

$$CD_N = D_{KL}(P_0 \| P_\infty) - D_{KL}(P_N \| P_\infty) \geq 0. \tag{2.34}$$

[1] Hinton [2002] used $N = 1$ in his original paper.

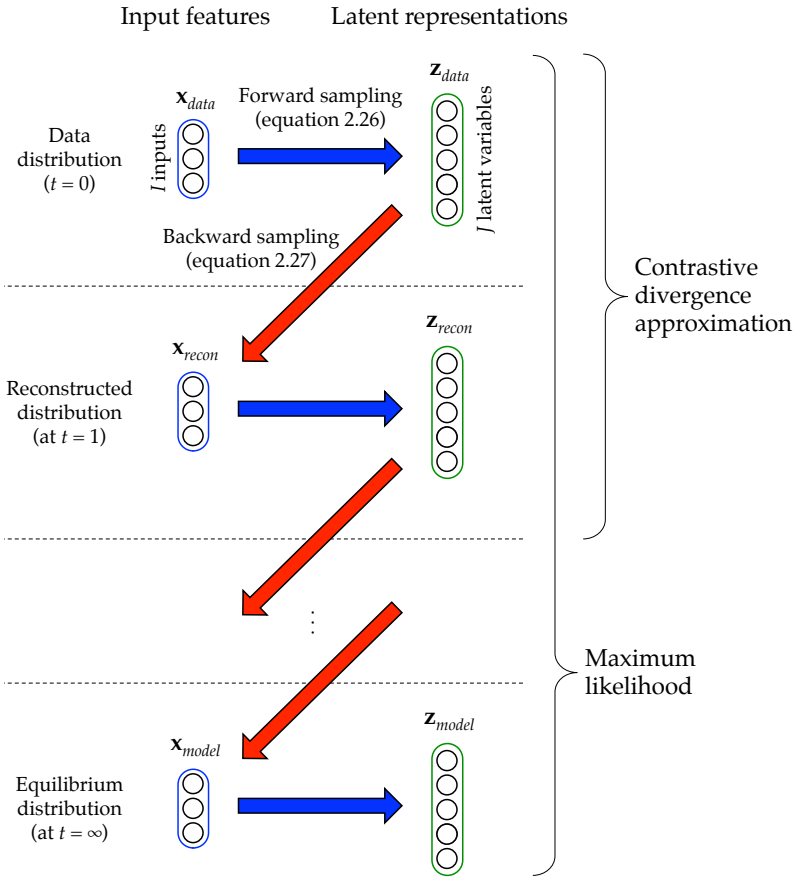


FIGURE 2.9: Gibbs sampling relaxation of contrastive divergence learning. The maximum likelihood of the data distribution can be obtained by performing alternating (forward-backward) Gibbs sampling between the input layer $\mathbf{x}$ and latent layer $\mathbf{z}$, from the data distribution samples (at $t = 0$) until $t = \infty$. Contrastive divergence approximates the equilibrium distribution using reconstructed samples from a small finite number of sampling steps. One sampling step ($N = 1$) is used in this example.

The partial derivative with respect to the parameters is

$$\begin{aligned}
\frac{\partial CD_n}{\partial w_{ij}} &= \frac{\partial D_{KL}(P_0 \| P_\infty)}{\partial w_{ij}} - \frac{\partial D_{KL}(P_N \| P_\infty)}{\partial w_{ij}} \\
&\approx -\left\langle \frac{\partial \log \sum_{\mathbf{z}} \exp(-E(\mathbf{x}, \mathbf{z}))}{\partial w_{ij}} \right\rangle_{data} + \left\langle \frac{\partial \log \sum_{\mathbf{z}} \exp(-E(\mathbf{x}, \mathbf{z}))}{\partial w_{ij}} \right\rangle_{recon} \\
&\approx \left\langle \frac{\partial E(\mathbf{x}, \mathbf{z})}{\partial w_{ij}} \right\rangle_{data} - \left\langle \frac{\partial E(\mathbf{x}, \mathbf{z})}{\partial w_{ij}} \right\rangle_{recon} \\
&\approx \langle x_i z_j \rangle_{data} - \langle x_i z_j \rangle_{recon}, \qquad (2.35)
\end{aligned}$$

where $\langle \cdot \rangle_{recon}$ is the expection of the reconstructed states after N sampling steps. This results in the following approximate gradient descent:

$$w_{ij} := w_{ij} + \varepsilon \left(\langle x_i z_j \rangle_{data} - \langle x_i z_j \rangle_{recon} \right), \qquad (2.36)$$

where the energy of samples from the data distribution is decreased, while raising the energy of reconstructed states that the network prefers to real data. Algorithm 2.2

Algorithm 2.2: RBM training with contrastive divergence

```
Initialize W
repeat
    Get X_0 from randomized training batch
    Sample P_0(Z_0|X_0)                                   // Equation 2.26
    for n = 1 to N do                          // Alternating Gibbs sampling
        Sample P_n(X_n|Z_{n-1})                           // Equation 2.27
        Sample P_n(Z_n|X_n)                               // Equation 2.26
    end
    Update w_ij := w_ij + Δw_ij                           // Equation 2.36
until convergence
```

presents the iterative procedure for training an RBM with a N Gibbs sampling step followed by parameter updating.

CD_1 is the fastest since it requires the minimum number of sampling step. It generally works well for most purposes. Although larger samples of N produce estimates that are closer to the true likelihood gradient, they need more time to compute and may result in high estimator variances. To avoid unnecessary sampling noise [Hinton, 2010] and reduce the variance of the estimator [Swersky et al., 2010], Rao-Blackwellization [Blackwell, 1947] is often employed, where w_{ij} is updated using activation probabilities $P(x_i|\mathbf{z})$ and $P(z_j|\mathbf{x})$, instead of their binary states. A variant of the contrastive divergence learning algorithm, known as persistent contrastive divergence [Neal, 1992; Tieleman, 2008], which is a stochastic approximation procedure [Robbins and Monro, 1951; Younes, 1989, 1999] that runs the Markov chain from a persistent state rather than running a new Gibbs sampling chain for every training example. An alternative to Gibbs sampling for generating the density model is to use a tempered Markov Chain Monte Carlo [Desjardins et al., 2010].

Variants of restricted Boltzmann machines. There are several variants of the RBMs, focusing on different aspects of the model. Rather than using stochastic binary units, an obvious extension is to use continuous units [Chen and Murray, 2003], softmax activated units assuming multiple discrete values, Gaussian units with stochastic Gaussian noise [Hinton and Salakhutdinov, 2006; Lee et al., 2009], binomial units, Poisson units, Student's t-distributed units [Welling et al., 2003] and rectified linear units. In general, any type of unit within the exponential family may be used [Welling et al., 2005]. However, such RBMs are typically more difficult to train in practice than binary RBMs. Please refer to Hinton [2010] for more details regarding the practical implementation of RBMs.

Some go beyond modeling single distributions of the latent variables using more sophisticated RBMs, such as the mean-covariance RBMs [Ranzato and Hinton, 2010;

Dahl et al., 2010], spike-slab RBMs [Courville et al., 2011], and the gated RBMs [Memisevic and Hinton, 2007], which can be extended to the factored three-way [Taylor and Hinton, 2009] and higher-order models [Memisevic and Hinton, 2010]. These models generally work on a more complicated energy function [Sejnowski, 1987]. In addition, the generative learning of the RBM could be coupled with discriminative learning [Larochelle and Bengio, 2008]. Other variants of the RBM that focus on image modeling, such as the convolutional RBM [Lee et al., 2009; Norouzi et al., 2009], will be discussed in detail in Section 2.3.3.

Selectivity regularization. Just as it is common to regularize the coding of decoder networks with sparsity, practitioners of RBMs tend to choose selectivity as the main regularizer to complement maximum likelihood approximation and learn interesting representations. Selectivity is the measure of the activity of a latent variable across a set of instances. Typically, the objective of the regularizer is to generate a coding that have low-average activation across the examples during training [Lee et al., 2008; Nair and Hinton, 2009; Hinton, 2010]. Section 3.2 provides a detailed review of these existing regularization methods.

2.3 Learning deep representations

A hierarchical architecture consists of multiple layers combined from series of basic operations. The architecture takes raw input data at the lowest level and processes them via a sequence of basic computational units until the data is transformed to a suitable representation in the higher layers to perform task, such as classification. Multiple layers of distributed coding allow the network to encode highly varying functions efficiently [Bengio, 2009]. An architecture with four or more representational layers is considered to be a deep architecture [Hinton et al., 2006; Bengio et al., 2006]. The learning of deep architectures has emerged, as an effective framework for modeling complex relationships among high-dimensional data and discovering higher-level abstractions. It learns a hierarchy of meaningful representations that carry some intrinsic value for classification tasks. As a result, deep learning methods have been suitably and successfully employed for a variety of problems, in domains like computer vision, audio processing and language understanding.

The current methods for learning deep architectures are a cumulation of much research over the years. Neural networks, popularized in the 1980s, revolutionized the notion of learning distributed representations from data. In particular, fully-connected multi-layered perceptrons, having been shown to be universal approximators, can represent any function with its parameters [Hornik, 1991]. However, researchers often have to deal with the problems of a huge number of parameters and the difficult

non-convex optimization problem, which can become tedious and difficult to manage for deep networks [Bengio, 2009]. Recent algorithmic developments in unsupervised feature learning together with the ever increasing computational power of machines, enabled such deep networks to be trained.

This section discusses the main motivations and strategies for learning deep architectures and introduces three families of deep architectures and an adaption scheme to model image data using the convolution operator.

2.3.1 Motivations and strategies

A typical deep architecture takes raw input data at the lowest level and processes them via a sequence of basic computational modules, such as connectionist models (Section 2.2), until the data is transformed to a suitable representation in the higher layers.

Motivations. The main motivation for learning deep architectures is to discover abstraction from the lowest-level features to the highest-level concepts, either automatically or with as little human intervention as possible. The objective is to learn a hierarchy of representations, such that the representations at higher-levels are composed of lower-level ones. The deep hierarchical structure of the architecture is analogous to the multiple levels of abstraction that humans naturally describe the visual world [Bengio, 2009]. This multiple level structure also corresponds to the organization of neuronal encoding by the visual system in our brains [Karklin and Lewicki, 2008; Lee et al., 2008]. Besides these, the computational resources for training and inference should scale well with data cardinality and dimensionality, and the model should remain robust on noisy data.

From a representational point of view, a deep, yet compact, architecture is able to model complex functions more efficiently than a shallower architectures. Particularly, if a function is compactly represented by L layers, it may require an exponential number of units to represent the same function using only $L-1$ layers [Bengio, 2009]. As a result, if an insufficiently deep architecture is used to model a complex function, the architecture might be very large to compensate for the loss in representational flexibility as compared to a deeper one. Just like their shallower cousins, deep networks have been shown to be universal function approximators [Sutskever and Hinton, 2008; Le Roux and Bengio, 2010].

General strategy. A deep architecture has an extensive parameter space, whereby searching for an ideal solution is challenging due to the non-convex supervised optimization problem and multiple local minima that depends on parameter initialization [Erhan et al., 2009]. This problem intensifies when the number of layers in the architecture increases. An effective strategy to train deep architectures remained illusive until the seminal work by Hinton et al. [2006], which was quickly followed by similar findings by Ranzato et al. [2006] and Bengio et al. [2006]. The methods follow a similar general approach, which first considers each layer as an unsupervised module and stacking them in a greedy-layer wise manner. Subsequently, a supervised fine-tuning step can be performed to optimize the architecture for the required task.

Greedy unsupervised learning. A key component for learning deep networks is to perform unsupervised learning in a greedy layer-wise manner [Hinton et al., 2006; Ranzato et al., 2006; Bengio et al., 2006]. Shallow unsupervised modules, such as those described in Sections 2.2.2 and 2.2.3, are exploited by repeatedly training and stacking them individually, one layer at a time from the bottom-up. When a new layer is stacked above the existing network, the parameters bridging the existing representation with a new layer is trained locally within the model using the previous layer as its inputs. Stacking layers encourages higher-level abstractions of the input data within the internal representations of the deep architecture. Given a training dataset $\mathcal{D}_{train}$, the procedure of this greedy learning algorithm to train a deep network with L unsupervised layers is described in Algorithm 2.3.

Algorithm 2.3: Unsupervised greedy layer-wise learning

```
1 Initialize input data Z_1 ← D_train
2 for l = 1 to L − 1 do
3     Initialize W_l
4     repeat
5         Update W_l by learning from Z_l          // Upsupervised learning
6     until convergence
7     Fix W_l and generate samples for Z_{l+1}
8 end
```

This greedy layer-wise unsupervised strategy is usually treated as a pre-training step before supervised (usually discriminative) learning. In the context of vision, before the model can perform recognition, it should first be able to generate input images [Hinton, 2007b]. Unsupervised pre-training helps the overall optimization by initializing the parameters around a good local minimum Larochelle et al. [2009], easing the difficult deep supervised optimization problem. This has a regularization effect for subsequent supervised training by improving generalization [Erhan et al., 2009, 2010].

Supervised fine-tuning. While unsupervised learning is effective in modeling the distribution of the input data, the representations learned may not be perfectly aligned with a given classification problem. One can imagine that the pre-training phase initializes a set of data in the same manner, regardless of how the classes are defined within the dataset. As such, supervised learning is crucial to fine-tune the parameters of the model and to encourage a binding to the classification task.

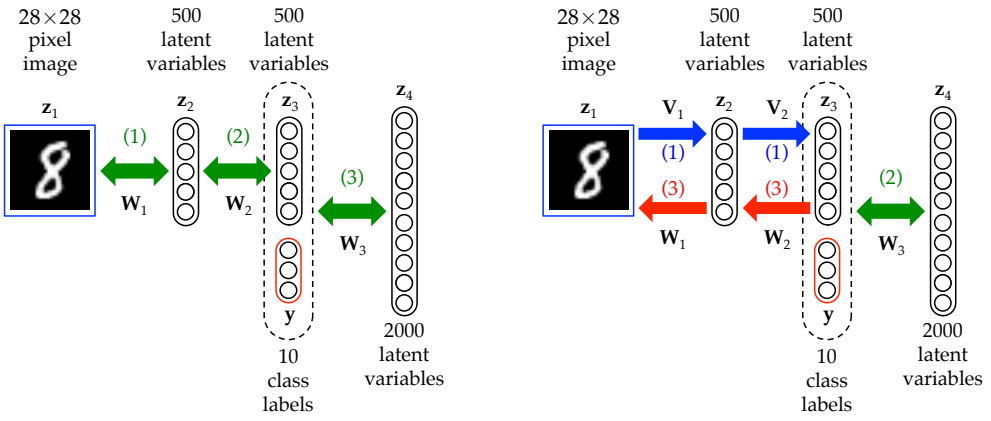
While unsupervised pre-training is performed in a greedy layer-wise manner, supervised learning globally optimizes the entire network. A popular method to introduce the labeled information to a deep network is through discriminative learning using the error backpropagation algorithm [Hinton and Salakhutdinov, 2006]. This is, essentially, the same algorithm as that for training multilayer perceptrons, as described in Section 2.2.1. Additionally, generative fine-tuning can serve as an alternative to discriminative learning by using a up-down back-fitting algorithm [Hinton et al., 2006].

2.3.2 Fully-connected deep architectures

There are three main families of deep architectures that originate from the various shallow unsupervised learning modules with fully-connected structure. They are namely: 1) deep belief networks [Hinton et al., 2006], 2) deep decoder-based networks [Ranzato et al., 2006] and 3) deep auto-encoder networks [Bengio et al., 2006]. The networks follow a similar general strategy of training by using greedy unsupervised learning followed by supervised fine-tuning. The main difference between the architectures is the choice of the basic building block. Deep learning is not restricted to connectionist models [Bengio, 2009; Deng and Yu, 2011].

Deep belief networks. Deep belief networks (DBN) [Hinton et al., 2006] are probabilistic graphical models made up of a hierarchy of stochastic latent variables using restricted Boltzmann machines (RBM) [Smolensky, 1986] as the basic building block. DBNs have also been shown to be universal approximators [Le Roux and Bengio, 2008] and are able to eliminate the notion of explaining away during inference [Wellman and Henrion, 1993; Hinton et al., 2006]. A variant of the DBN is the deep Boltzmann machine [Salakhutdinov and Hinton, 2009, 2012], which disregards the restricted bipartite connectivity of the RBM structure (see Figure 2.7).

DBNs have been applied to various problems such as handwritten digit recognition [Hinton et al., 2006], video and motion sequence recognition [Sutskever and Hinton, 2007; Taylor and Hinton, 2009], dimension reduction [Hinton and Salakhutdinov, 2006], and data indexing [Salakhutdinov and Hinton, 2009]. Figure 2.10 shows a DBN for handwritten digit recognition during the two phases of training.



(a) Unsupervised greedy layer-wise learning (b) Supervised fine-tuning (up-down algorithm)

FIGURE 2.10: Deep belief network for handwritten digit recognition. (a) Unsupervised learning is performed one layer at a time from the bottom up. (b) Supervised fine-tuning first unties forward and backward weights of the bottom two RBMs. A bottom-up pass learns to reconstruct input data, while a top-down pass learns to associate the model to the topmost layer.

For unsupervised pre-training (Figure 2.10(a)), the parameters for every consecutive pair of representational layers are learned as an RBM. An RBM stack is trained greedily from the bottom-up, with the latent activations of each layer used as the inputs for the next RBM. As a result, each layer captures higher-order correlations of the bottom-layer. Each RBM approximates the likelihood of its input distribution through an energy function (see Section 2.2.3). Hinton et al. [2006] showed that when higher-level layers are sufficiently large, the variational bound on the likelihood always improves with depth. Two methods have been proposed for supervised fine-tuning.

The first method uses a up-down back-fitting algorithm [Hinton et al., 2006], a variant of the "wake-sleep" algorithm [Hinton et al., 1995]. The algorithm is initialized by untying the recognition $\mathbf{W}_l$ and generative $\mathbf{V}_l$ weights. Figure 2.10(b) illustrates an example of the training procedure of this algorithm. First, a stochastic bottom-up pass is performed to obtain $\mathbf{z}_2$ and $\mathbf{z}_3$. The top-down generative weights $\{\mathbf{V}_1, \mathbf{V}_2\}$ are adjusted to be good at reconstructing the layer below by maximizing the input likelihood. Next, a few iterations of alternating sampling are done at the top-level RBM between $\mathbf{z}_4$ and the concatenated vector $[\mathbf{z}_3, \mathbf{y}]$. Using contrastive divergence the RBM is updated by fitting to its posterior distribution. Finally, a stochastic top-down pass adjusts bottom-up recognition weights $\{\mathbf{W}_1, \mathbf{W}_2\}$ to reconstruct the activations of the layer above.

The second method is more straightforward and uses the error backpropagation algorithm to fine-tune the parameters. It has been shown to perform well when initialized by first learning a model of input data using unsupervised pre-training [Hinton and Salakhutdinov, 2006].

Deep decoder-based networks. Just like a DBN can be formed by stacking several RBMs in a hierarchy, decoder networks (Section 2.2.2) can be treated as unsupervised building blocks and stacked to become a deep network [Ranzato et al., 2006], as illustrated in Figure 2.11. As inspired by the DBN, a greedy unsupervised learning algorithm is used to learn the parameters one layer at a time. Each decoder network can be trained layer-by-layer from the bottom-up to form the deep network. New coding layers are optimized by learning to reconstruct the codings of the previous layer. This network, however, has its limitations during inference. Code optimization needs to be performed one layer at a time, resulting in slow inference.

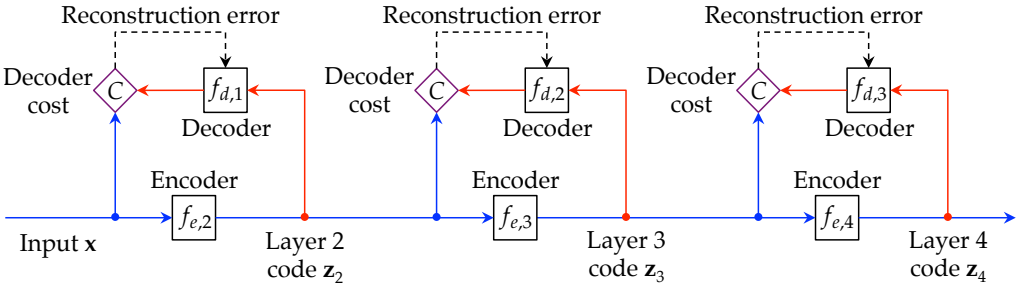


Figure 2.11: A deep decoder network. The network is composed of a hierarchy of decoder networks, each optimized by trying to optimize the reconstructions of the previous layer. During inference, the coding needs to be optimized for every layer.

A deep encoder-decoder network is a multilayer extension of the encoder-decoder network. Using the encoder-decoder network (Section 2.2.2) as the basic building block, the networks are stacked from the bottom-up to form the deep network (Figure 2.12). Learning a new layer requires performing the alternating optimization between the codes and the parameters, where a decoder and encoder are concurrently learned. As a result, the output of the network is generated simply by using the chain of encoders in a feedforward manner. This results in faster inference as compared to the deep network built from only decoder networks.

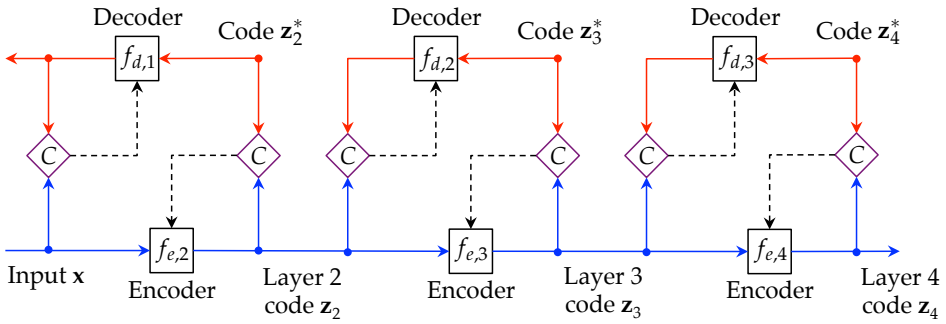


Figure 2.12: Deep encoder-decoder network. The network comprises of encoder-decoder building blocks stacked in a layer-wise hierarchy. Each new layer is learned in an unsupervised manner from the outputs of the previous layer. During inference, the chain of encoders directly produce the coding for the network.

Deep auto-encoder networks. Another similar deep architecture is the deep auto-encoder network (Figure 2.13) [Larochelle et al., 2009]. Instead of RBMs or decoder-based networks, the deep auto-encoder network is formed by a series of auto-encoder networks (Section 2.2.2), stacked above each other in a feature hierarchy. Each auto-encoder tries to minimize the reconstruction errors of the previous layer [Bengio et al., 2006; Vincent et al., 2008]. Special consideration needs to be put in place to avoid learning trivial solutions, such as the identity function. Also, the auto-encoder is not as good as the DBN in handling random noise in the input data [Larochelle et al., 2007].

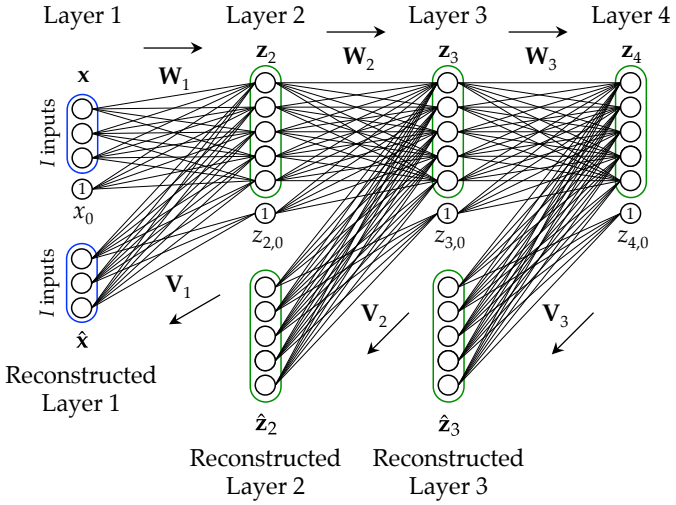


FIGURE 2.13: Deep auto-encoder network. The network learns a sequence of encoders that attempt to reconstruct the previous layer via a second set of weights.

2.3.3 Local connectivity and convolutional networks

In general, the fully-connected deep architectures presented above perform extremely well on modeling input data. However, due to the fully-connected structure between layers, it is difficult to scale up to handle high dimensional inputs, such as images. The images that these deep networks typically model only have a few hundred dimensions. For example, each image in the MNIST handwritten digit dataset [LeCun et al., 1998] is 28×28 pixels in size, resulting in a 784 dimensional input. An image with a size of 300×200 pixels, already has $60,000$ dimensions. Furthermore, modern digital camera sensors are capable of producing images with tens of millions of pixels.

A dominant approach for modeling large images is through the use of local connectivity as inspired by the receptive field scheme proposed by Hubel and Wiesel [1962]. The ideas were incorporated into an early hierarchical multilayer neural network known as the Neocognitron by Fukushima [1980]. The main concept is that parameters of the model are shared across various local patches of an image, resulting in invariance to spatial translation. In the 1980s and 1990s, LeCun et al. [1989, 1998] developed the

convolutional neural network that can be trained using error backpropagation. It focused on tackling the vision problem through a fully-supervised multilayer network with convolution operators in each layer mapping their inputs to produce a new representation via a bank of filters. The model produced exceptional performances for vision problems, such as handwritten digit recognition [LeCun et al., 1989, 1998].

The convolutional operator naturally shares the parameters of the model by coding the image locally across various spatial locations in the image. This helps to scale the model for large images, while taking into account spatial correlations in the image. In addition, a sub-sampling operator is sometimes used to perform pooling in the filtered representation and reduces the spatially dimensionality of the representation. Figure 2.14 shows an example of such a convolutional network with convolution and sub-sampling performed successively. We shall see later that this approach of modeling images is not much different from other computer vision models (Section 2.4.1).

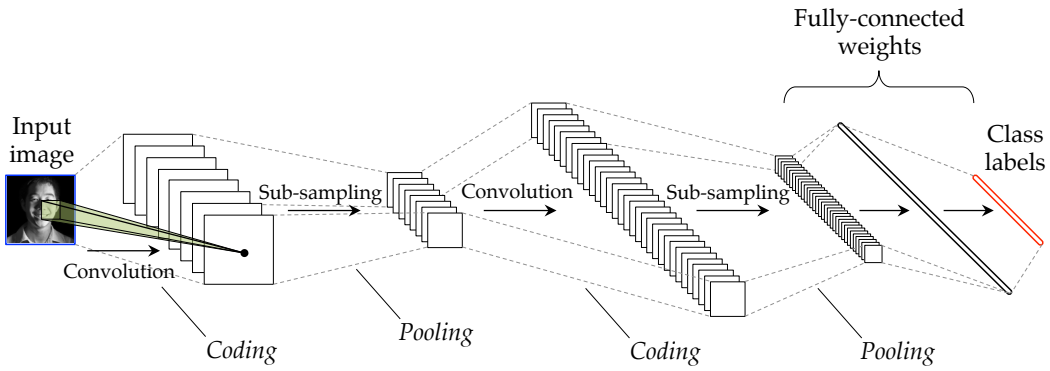


Figure 2.14: Convolutional network. The network typically consists multiple layers of convolutional and pooling operators to transform an input image to a high dimensional vector, which can be projected via fully connected weights to an output vector.

Such a highly adapted hierarchical local connectivity has the potential to encode structure suitable for modeling images, such that even with random weights in the early layer, performance remains impressive [Ranzato et al., 2006]. Although a fully trained convolutional network would still produce better performances, the challenge of performing gradient descent on the deep network is slightly reduced due to the local connectivity of the network, resulting in a smaller fan-in. This allows the error signal to propagate through more layers [Bengio, 2009]. However, being fully-supervised, the challenge will be to avoid getting stuck in a poor local minimal.

Most recently, convolutional deep neural networks [Krizhevsky et al., 2012] have recently emerged as the best performing model in the *ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)* using a large-scale dataset with $1,000,000$ labeled training images. The network consists of seven learned layers (five convolutional and two fully-connected) that map image pixels to the semantic-level. The model is trained

in an entirely supervised manner using the stochastic gradient descent method with the backpropagation algorithm. Being fully-supervised, this network requires a lot of labeled training data to perform well and avoid over fitting. Despite the good performances on the large-scale dataset, it has not yet been shown to be able to learn meaningful representations for classifying moderate-sized datasets, with relatively fewer labeled training examples.

With the emergence of the recent foundational deep learning methods, the computer vision community now has a new set of tools to apply. Due to the representational depth and learning plasticity of these networks, the variety of information it can learn to represent is extensive. Norouzi et al. [2009] and Lee et al. [2009] exploited the learning capabilities of stacks of restricted Boltzmann machines by adding convolutional operators to learn to classify large images from the pixel-level. However, image classification results fell short of the state-of-the-art performances by the vision-based bag-of-words model (Section 2.4). Tiled convolutional approaches [Ranzato et al., 2010; Le et al., 2010] had also been suggested to tackle the problem of high spatial dimensionality, but did not result in a significant breakthrough in image classification.

2.4 Modeling images using visual words

To bridge the semantic gap, an image classification model transforms images from the pixel level to intermediate representations before classification takes place. These representations exist in a variety of levels, from low-level image pixels and feature descriptors, to mid-level visual coding and high-level semantic concepts. A low-level representation describes the perceptual appearance of an image. As it is transformed to a mid-level representation, invariance to visual transformations may be incorporated. The objective of the mid-level representation is to integrate the low-level representation, improve class-based discrimination and construct a single vectorial representation for classification into high-level concepts or classes. One of the main goals of this thesis is to improve the mid-level representations by using a visual word model that maps between the low-level and high-level representations.

2.4.1 Pipeline of the bag-of-words model

A multi-step model known as the bag-of-words (BoW) model, which was inspired from textual information retrieval [Salton and McGill, 1983], was adopted for content-based image retrieval (CBIR) through the quantization of color [Ma and Manjunath, 1997] and Garbor feature vectors [Fournier et al., 2001]. Subsequently formalized for the SIFT descriptor by Sivic and Zisserman [2003], the objective of the model is to represent an image as a bag of visual words or features.

To classify an image from its pixels into its semantic category, the model performs a series of four consecutive steps, namely: 1) feature extraction, 2) feature coding, 3) pooling, and 4) classification. The pipeline of the four steps of the BoW model is illustrated in Figure 2.15. This pipeline results in the state-of-art performances on many image classification problems, especially for object [Fei-Fei et al., 2004; Griffin et al., 2007] and scene [Lazebnik et al., 2006] images.

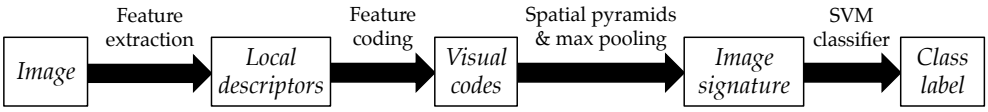


Figure 2.15: Pipeline of the bag-of-words model. Four data transformation steps are used to map an image from the pixel level to its semantic class.

In a typical setup, gradient-based local image descriptors, such as scale-invariant feature transform (SIFT) [Lowe, 1999], are used to describe an image. They are discriminative yet robust to various image transformations. A crucial aspect of the BoW model is the transformation from the set of local descriptors to a constant-sized image vector used for classification. Converting the set of local descriptors into the final vectorial image representation is performed by a succession of two steps: feature coding and pooling. The feature coding step embeds the local descriptor representation into a higher-dimensional feature space of greater complexity and potentially improves class-wise separability. The pooling step combines local representations into a single vector used for classification using support vector machines (SVM) [Vapnik, 1995]. A popular method for pooling is the spatial pyramid [Lazebnik et al., 2006] with max-pooling [Boureau et al., 2010a].

The combination of SIFT, spatial pyramids [Lazebnik et al., 2006] with max-pooling [Boureau et al., 2010a] and SVMs [Vapnik, 1995] is a proven strategy for achieving good image classification performances, especially when linear classifiers are used Boureau et al. [2010a]. The main shortcoming of the BoW model is that feature coding is generally a fixed (non-learned) and flat (single-layer) operation. The flat structure limits the representational power of the model, while the lack of learning makes it difficult to adapt to different data and vision tasks. Learning a good visual dictionary to perform feature coding will be one of the main subject of this thesis.

Relation with convolutional networks. The SIFT descriptor can be perceived as a two-stage operation of local gradient coding and spatial pooling (Section 2.4.2). As a result, the BoW model, when seen in its entirety in Figure 2.16, corresponds closely to the convolutional model described in Section 2.3.3, although there are some structural differences betweeen the architectures. Essentially, the BoW model executes a sequence of two coding-pooling operations to form the image signature – a single vector describing the entire image – used for classification.

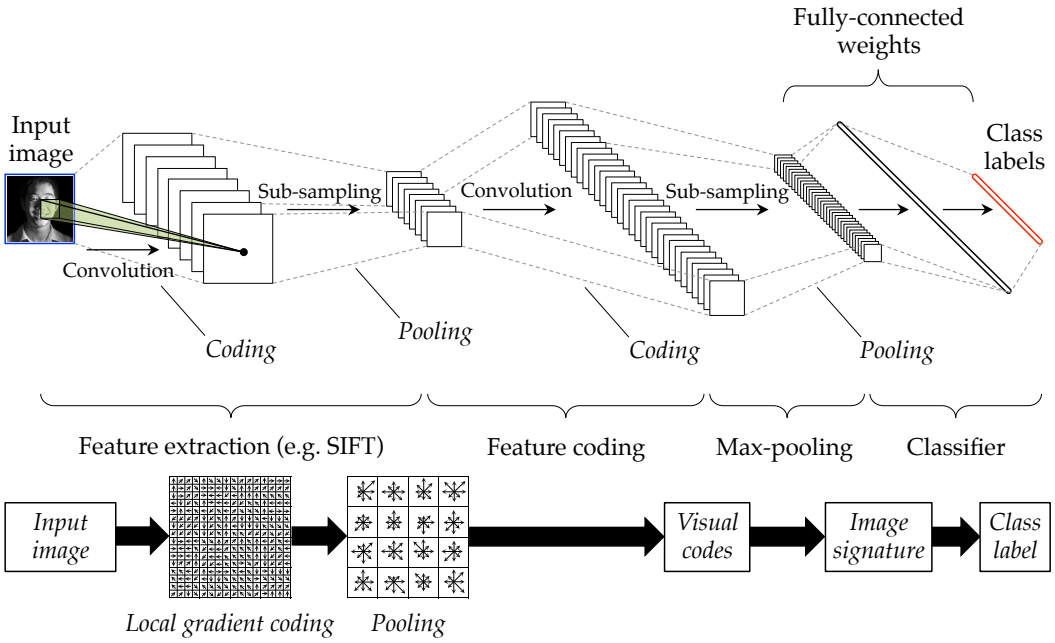


FIGURE 2.16: Relating the bag-of-words model to convolutional networks. The four-step pipeline of the bag-of-words model (below) that uses SIFT descriptors resembles a convolutional network (above) with two successive convolution and sub-sampling steps, followed by a fully connected structure.

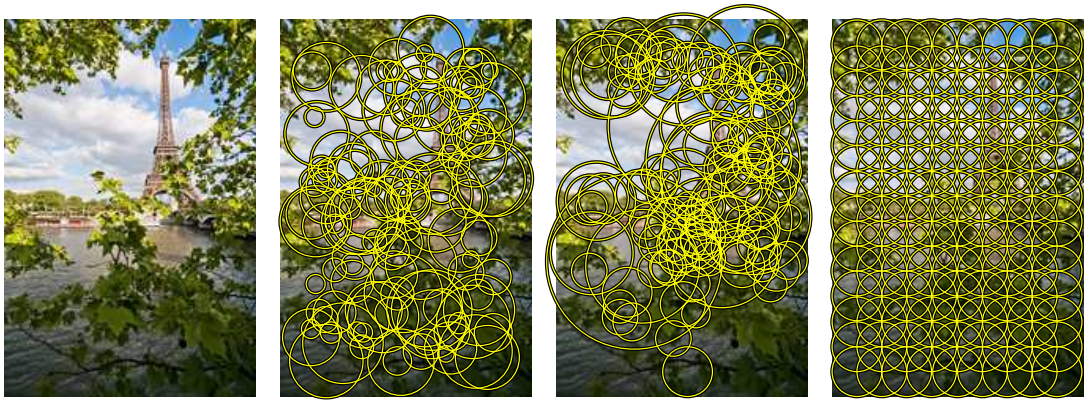
2.4.2 Local feature extraction

Extracting low-level image descriptors is the first important step for many computer vision models. In general, there are two approaches to characterize an image using descriptors: globally and locally. The difference between the two is the region over which they represent. A single global descriptor is employed to model the statistics of the entire image, while a set of multiple local descriptors is extracted from the image, with each descriptor representing only a small portion or patch of the image.

Various global descriptors have been proposed to represent pixels of an image without the need for any form of image segmentation. Examples include the color histogram [Swain and Ballard, 1990, 1991], the collection of views [Franz et al., 1998; Wolf et al., 2002] and the GIST descriptor [Oliva and Torralba, 2001; Torralba, 2008], have been used for various vision applications, such as scene recognition and indexing. However, they are not invariant to image and object transformations and are ineffective for recognizing images with clutter [Tuytelaars and Mikolajczyk, 2008].

Local descriptors, on the other hand, only describe a small part of the image and are collectively able to overcome the limitations of global descriptors. However, the challenge of local descriptors arise from the need to integrate the image information extracted from various locations in the image to perform image classification. This is where the pooling step in the BoW model is put to use.

Local sampling strategies. Relating to the region a descriptor represents, is the location whereby they are sampled from. For local descriptors, there are generally three sampling choices: random sampling, keypoint sampling or dense sampling [Nowak et al., 2006], as illustrated in Figure 2.17. Random sampling randomly selects locations and scales from which descriptors are computed. Keypoint sampling focuses on extracting descriptors from salient locations, such as edges, corners and blobs, to promote invariance to image transformations, illumination changes and noise for object matching [Schmid et al., 2000]. Dense sampling, on the other hand, is able to cover the entire image by extracting descriptors uniformly across the image using a uniform grid, which may be overlapping and have multiple scales, without bias toward any specific portion of the image.



(a) Input image. (b) Random sampling. (c) Keypoint sampling. (d) Dense sampling.

Figure 2.17: Local descriptor sampling. Local descriptors can be sampled from an image either randomly, using keypoints or densely across the image.

Due to its position at the start of the image classification pipeline, the resulting performance is highly biased towards the choice of sampling method used. Empirically, keypoint sampling suffers from a loss in discriminative power for classification, as compared to dense sampling [Jurie and Triggs, 2005] and even against random sampling [Nowak et al., 2006]. Dense sampling has been successfully employed on numerous occasions for various image classification problems [Chatfield et al., 2011; Law et al., 2012].

Descriptor representations. At the lowest level, the model describes the image using a set of robust of gradient-based local descriptors, such as the scale-invariant feature transform (SIFT) [Lowe, 1999], the speeded-up robust features (SURF) [Bay et al., 2008] and the histogram of orientated gradients (HOG) [Dalal and Triggs, 2005]. These descriptors are invariant to various image transformations, such as geometric

and photometric transformations, which are essential when addressing image classification problems. In this thesis, the SIFT descriptor is used in conjunction with the BoW model, which motivates the following expanded discussion.

Scale-invariant feature transform (SIFT). The SIFT descriptor [Lowe, 1999] is one of the most popular descriptors used for a variety of vision problems, from image classification to 3D modeling and video tracking. When originally proposed, the SIFT descriptor was used in conjunction with keypoint sampling based on the difference-of-Gaussian (DoG) detector. The descriptor has since been employed separately to characterize a local image patch. It is known to exhibit invariance towards common image deformations such as scale, rotation and affine transformations, and changes in illumination [Mikolajczyk and Schmid, 2005].

Figure 2.18 illustrates the process of computing the SIFT descriptor for a given image sample point and scale. Local gradient magnitude and orientation are first computed for all spatial positions in a grayscale image patch. They are then weighted by a Gaussian window centered in the middle of the descriptor window. The patch is then partitioned into a 4×4 grid. Local image gradient information is *coded* into 8 possible orientation bins and *pooled* into a histogram within each spatial grid. The 8 coded orientations in the $4 \times 4 = 16$ spatial grids combine to form the 128 dimensional SIFT descriptor.

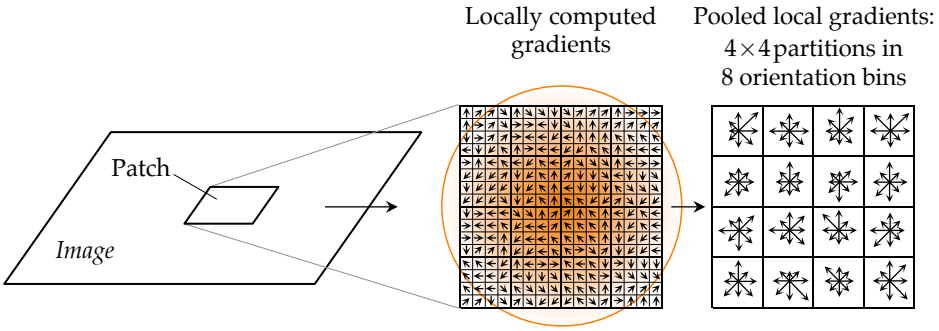


FIGURE 2.18: Scale-invariant feature transform (SIFT) descriptor. SIFT descriptors are extracted from image patches sampled from the image. Local gradients are computed and weighted using a Gaussian window (indicated by the shaded circle). From a 4×4 spatial partition, histograms with 8 orientation bins are computed from the extracted gradients within each partition, resulting in the 128 dimensional SIFT vector.

There are many variants to the SIFT descriptor. Conventionally, the SIFT descriptor is computed from grayscale patches. The most obvious way to extend the descriptor is to incorporate color information by computing image gradients over the color channels of different color spaces, such as the red-green-blue (RGB) space, hue-saturation-value (HSV) space and color-opponent space. The descriptor for each channel is then concatenated to form the final local descriptor [Van de Weijer and Schmid, 2006; Burghouts and Geusebroek, 2009; Van de Sande et al., 2010].

Another family of variants builds upon the idea of computing gradients but modifies the spatial partitioning over which the histograms are computed. Some examples are as follows. The PCA-SIFT does not partition the image patch, but perform dimension reduction on the extracted gradients by using principle component analysis (PCA) [Ke and Sukthankar, 2004]. This may possibly result in a smaller descriptor. Meanwhile, the rotation invariant feature transform (RIFT) descriptor computes the histograms over concentric rings, to enhance in-plane rotational invariance [Lazebnik et al., 2004]. Finally, the gradient location-orientation histogram (GLOH) descriptor uses a log-polar grid to compute the descriptor and subsequently employs PCA to compress the dimensionality of the descriptor [Mikolajczyk and Schmid, 2005]. This attempt to post-process the local descriptor after it has been computed by mapping it into another feature space leads to the coming discussion on feature coding (Section 2.4.3).

2.4.3 Learning visual dictionaries for feature coding

In the BoW model (Figure 2.15), the feature coding step transforms the local image descriptors into a representation of visual codes. Due to visual word ambiguity [van Gemert et al., 2010], this coding step has garnered much attention by those trying to capture meaningful visual representations. Using a visual dictionary with J visual words, the BoW model describes an image as a histogram of visual word occurrences. More generally, the visual dictionary contains visual words that help project local descriptors to another feature space for the subsequent steps in the BoW pipeline.

However, there is neither a clearly defined dictionary to relate visual representations to semantics, nor a well established syntax to combine such visual representations. As a result, many researchers focus their efforts on applying statistical learning techniques to find a good visual dictionary from local descriptors extracted from a training dataset (Figure 2.19). This is one of the primary objectives of this thesis.

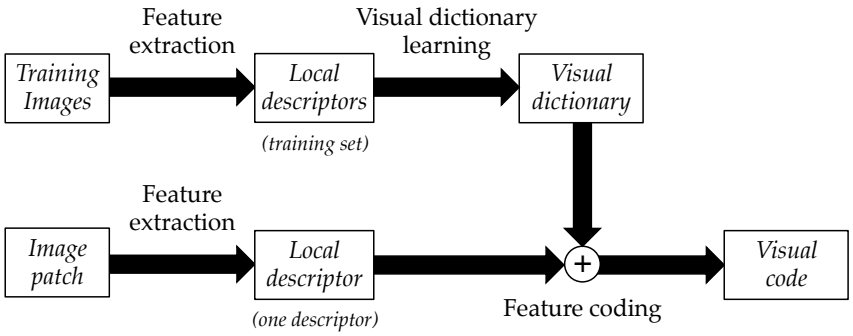


Figure 2.19: Learning a visual dictionary for feature coding. From a set of local descriptors extracted across images in the training dataset, a visual dictionary is learned to perform feature coding of individual local descriptors into visual codes.

Given an unordered set $\mathbf{X} = \{\mathbf{x_k} : k \in [1, K]\}$ of local descriptors $x_k \in \mathbb{R}^I$, which have been extracted from the image dataset, the objective of the feature coding step is to project $\mathbf{X}$ into a set $\mathbf{Z} = \{\mathbf{z_k} : k \in [1, K]\}$ of J dimensional visual code (Figure 2.20). The reference points for this mapping are collectively stored as visual dictionary $\mathbf{W}$ with a collection of J visual words $\mathbf{w}_j$. Formally, the projection from an image descriptor $\mathbf{x}_k$ to visual code z_{jk} can be defined using an encoding function:

$$z_{jk} = f_e(\mathbf{x}_k, \mathbf{w}_j). \tag{2.37}$$

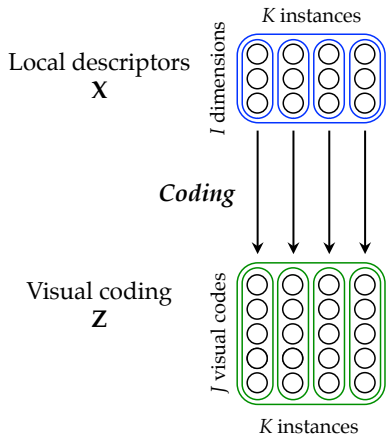


FIGURE 2.20: Visual feature coding. In this step, each of the K local image descriptors of I dimensions is projected to a visual code of J dimensions.

Hard and soft assignment coding. Sivic and Zisserman [2003] introduced an early attempt to learn a visual dictionary $\mathbf{W}$ from the SIFT descriptors in the training dataset by using the k-means clustering algorithm [Lloyd, 1982].[2] The main intuition for using k-means to generate the visual dictionary is to summarize the training examples in the input feature space by means using the cluster centers as reference points. Each cluster center corresponds to one visual word $\mathbf{w}_j \in \mathbb{R}^I$. If J cluster centers are used to generate the dictionary, each descriptor $\mathbf{x}_k$ can be encoded by assigning it to the nearest visual word:

$$z_{jk} = f_e(\mathbf{x}_k, \mathbf{W}) = \begin{cases} 1 & \text{if } j = \underset{\hat{\jmath} \in \{1,\dots,J\}}{\arg\min} \|\mathbf{x}_k - \mathbf{w}_{\hat{\jmath}}\|_2^2 \\ 0 & \text{otherwise.} \end{cases} \tag{2.38}$$

This method coding scheme is also known as hard assignment coding, because it performs a hard quantization based on the closest prototype, while disregarding the density and distribution of the visual words in the feature space. More generally, any distance metric may be used to decide on the visual word assignment. However, the distances need to be computed for as many times as there are visual words.

[2] k-means is similar to the Linde–Buzo–Gray vector quantization algorithm [Linde et al., 1980].

An alternative to hard assignment coding is the soft assignment coding scheme [van Gemert et al., 2010]. Soft assignment aims to reduce the quantization errors and ambiguity resulting from the hard quantization [Philbin et al., 2008]. Instead of a hard binary quantization, the representation can take the form of distances between the descriptors and visual words. A straightforward implementation is to employ the softmax function that converts the distances into soft assignment weights as follows:

$$z_{jk} = f_e(\mathbf{x}_k, \mathbf{W}) = \frac{\exp(-\beta \|\mathbf{x}_k - \mathbf{w}_j\|_2^2)}{\sum_{\hat{j}=1}^{J} \exp(-\beta \|\mathbf{x}_k - \mathbf{w}_{\hat{j}}\|_2^2)}, \tag{2.39}$$

where β controls the softness of the selection. When $\beta \to \infty$, the assignment is equivalent to the hard assignment scheme. On the other end of the spectrum, when $\beta = 0$, the coding is the concatenation of the normalized Euclidean distance between the descriptor and each visual word. This coding may result in dense codes that lack discrimination. An alternative is to use a hybrid assignment, known as semi-soft coding [Liu et al., 2011], that performs soft assignment coding only for a small subset of visual words that are nearest to the descriptor.

One weakness of the model is the reliance on the simplistic k-means clustering algorithm. The algorithm is not guaranteed to converge to a global optimum and may require a very long runtime to converge [Vattani, 2011]. Moreover, the dictionaries learned may not be ultimately effective. Even using randomly selected local descriptors as the representative visual words produces respectable results in comparison [Nowak et al., 2006; Viitaniemi and Laaksonen, 2008]. Other clustering strategies can be adopted to learn the visual dictionary. For example, early attempts include the use of Kohonen self-organizing maps [Kohonen, 1982] to generate the visual words [Fournier et al., 2001]. Subsequently, a variety of other techniques were also attempted, such as mean shift clustering [Jurie and Triggs, 2005], hierarchical clustering [Nister and Stewenius, 2006], co-clustering [Liu and Shah, 2007] and agglomerative clustering [Leibe et al., 2008].

In addition, rather than encoding the distance between a local descriptor and a visual word as a single scalar value, soft assignment coding can be extended by encoding the distance in vectorial form. One way to do this is to use Fisher vectors to capture addition higher-order statistics of the distances between the local descriptors and the visual words [Perronnin and Dance, 2007]. Other aggregate approaches include the vector of locally aggregated descriptors (VLAD) method [Jégou et al., 2010] and the super-vector method [Zhou et al., 2010]. However, these cause a huge inflation in the representation size, where an image representation dimensionality in the order of millions is not unexpected.

Sparse feature coding using connectionist models. There is growing interest in applying machine learning techniques to improve the visual dictionary. Based on the general formulation of the encoding function (Equation 2.37), a visual word $\mathbf{w}_j$ can be treated as parameters directing a projection of descriptor $\mathbf{x}_k$ to the visual code z_{jk}. This projection can be performed using connectionist models (Section 2.2), such as decoder networks and restricted Boltzmann machines (RBM). These methods have well established feature coding and parameter learning techniques as introduced in Sections 2.2.2 and 2.2.3.

Sparse decoder networks. Consider a local descriptor $\mathbf{x}_k \in \mathbb{R}^I$ and a projection matrix $\mathbf{W} \in \mathbb{R}^{I \times J}$ containing the J visual words $\mathbf{w}_j$. The sparse coding optimization of the decoder network (Equation 2.16) attempts to find the vector of linear projections $\mathbf{z}_k^* \in \mathbb{R}^J$ that explicitly minimizes the reconstruction error of the descriptor, along with a regularization term that promotes sparse solutions:

$$\mathbf{z}_k^* = \underset{\mathbf{z}_k}{\operatorname{argmin}} \|\mathbf{x}_k - \mathbf{W}\mathbf{z}_k\|_2^2 + \lambda \|\mathbf{z}_k\|_1 \,, \tag{2.40}$$

where λ is a regularization constant and the ℓ_1 norm is often used as a surrogate of the ℓ_0 norm, leading to a convex problem. This setup has been effectively implemented to produce sparse codes for modeling images and has achieved positive results. [Yang et al., 2009; Mairal, 2010; Boureau et al., 2010a; Kavukcuoglu et al., 2010]. In particular, under the BoW model, Yang et al. [2009] used an efficient sparse coding algorithm by Lee et al. [2007], while Boureau et al. [2010a] used the SPAM toolbox[3]. Extensions to this method exploit locality constraints in the feature space to encourage spatially local coding [Yu et al., 2009; Wang et al., 2010; Oliveira et al., 2012].

The decoder method makes it possible to learn the mapping of input descriptors to sparse representations. However, the optimization of Equation 2.40 needs to be solved for every descriptor. This makes inference very slow, especially when there are many descriptors or when the visual dictionary is large (see Section 2.2.2 for further discussion). There are various approximations that help alleviate the computational costs. For example, Yang et al. [2010a] limited the optimization to within a component of a mixture model. Meanwhile, Boureau et al. [2011] used a small dictionary for sparse coding and another for pooling or incorporating locality constraints proposed by Wang et al. [2010], but the resulting visual representation is large.

A more direct approach to avoid performing the heavy minimization step of Equation 2.40 during inference is to introduce an encoder that is concurrently learned [Kavukcuoglu et al., 2010], resulting in an encoder-decoder network (also discussed

[3] http://www.di.ens.fr/willow/SPAMS

in Section 2.2.2). For feature coding, the encoder simply takes the local descriptor and performs a feedforward activation based on its learned parameters.

Restricted Boltzmann machines. A restricted Boltzmann machine (RBM) [Smolensky, 1986] is a special type of encoder-decoder network [Ranzato et al., 2007], which gives faster inference than the decoder network due to the learned encoder. The model was introduced previously in Section 2.2.3. Applied to computer vision, Lee et al. [2009] modeled image pixel patches with a hierarchy of RBMs, which were regularized to obtain low average responses Lee et al. [2008]. Convolution and sub-sampling operators (Section 2.3.3) were used to perform spatial aggregation of representations and handle the size of the images. These methods generally learn representations directly from image pixels rather than local descriptors.

To encode SIFT descriptors, Sohn et al. [2011a] used Gaussian RBMs to learn representations from SIFT, but the overall architecture is heavy because Gaussian RBMs are relatively more difficult to train. There are also some drawbacks of the a low average responses regularization [Lee et al., 2009; Nair and Hinton, 2009], which will be discussed in detail in Section 3.2.3.

Supervised dictionaries. Another class of methods exploit labeled data to increase class separation through discriminative supervised loss functions, such as mutual information loss [Lazebnik and Raginsky, 2009]. The supervised learning methods may be classified with respect to the scale in which image labels are incorporated. Some methods directly use a discriminative criterion for each local descriptor [Mairal et al., 2008; Jiang et al., 2011]. However, an image label usually does not propagate to every local portion of the image, so these methods do not pose the exact image classification problem and are more likely to suffer from noise. Other methods associate labels to a pooled global image statistic [Yang et al., 2008, 2010b; Boureau et al., 2010a]. However, since the visual dictionary is meant to encode local descriptors, the information has to be "un-pooled" for dictionary optimization, making the methods complex and slow. For example, in Yang et al. [2010b], backpropagation has to be implemented stochastically, resulting in long training times.

2.4.4 Pooling

In the pooling step, the aim is to construct a single vectorial representation $\mathbf{v} \in \mathbb{R}^J$ from the set of local visual codes $\mathbf{Z} \in \mathbb{R}^{J \times K}$ across the whole image. This image signature has the same dimensionality across all the images of possibly different sizes and serves as a consistent representation that can be used for image classification. The

general pooling function, defined as

$$v_j = f_p(\mathbf{z}_j), \tag{2.41}$$

transforms each visual coding dimension across the local coding instances into a single scalar value (Figure 2.21). The concatenation of these scalar values $\mathbf{v} = [v_1, v_2, \ldots, v_J]^T$ forms the image signature for the final classification step.

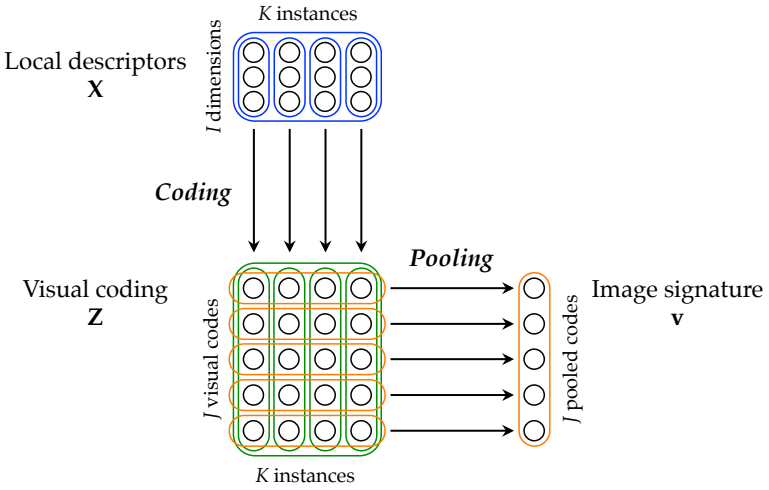


FIGURE 2.21: Visual feature coding and pooling. After the coding step, each visual coding dimension is described as a scalar value. The resulting vector is the signature of the image used for image classification.

In the classical pooling formulation of the BoW model, average (or sum) pooling is used to construct a histogram of occurrences of each visual coding dimension in the image. This operation can be formulated as

$$v_j = f_p(\mathbf{z}_j) = \frac{1}{K} \sum_{k=1}^{K} z_{jk}. \tag{2.42}$$

Average pooling has been extended for the BoW model to a max pooling operation [Yang et al., 2009]. Instead of diluting the coding by taking the average, max pooling selects the maximum value of each dimension:

$$v_j = f_p(\mathbf{z}_j) = \max_{k \in \{1,\ldots,K\}} z_{jk}, \tag{2.43}$$

which corresponds to an existential quantifier.

Boureau et al. [2010a] showed that max pooling outperforms average pooling, especially when linear classifiers are used. Boureau et al. [2010b] further suggested that the best form of pooling may be an operation between average and max pooling. From this, Feng et al. [2011] proposed the geometric ℓ_p-norm pooling, which attempts to learn the pooling parameters discriminatively. Zeiler and Fergus [2012,

2013] integrated the learning of a pooling operator that is specific to deep hierarchical networks for learning feature representations. While the above pooling methods output a scalar value, other work study the pooling beyond a scalar pooling method to capture higher-order statistics using vector-based pooling Avila et al. [2011, 2013].

Spatial pyramidal pooling. If the image is large or has many local descriptors, then pooling over the entire image may lead to significant loss in information due to the high fan-in of each pooled code. Instead of pooling directly over the entire image, the information loss can be reduced if the number of descriptors is reduced. For example, if pooling is performed on one descriptor, there will be no loss due to the pooling operation, regardless of the type of pooling employed - average or max. Inspired by Grauman and Darrell [2005], Lazebnik et al. [2006] proposed the spatial pyramid scheme to attenuate the loss of spatial information when pooling over the whole image. The pooling scheme independently first pools information from different image regions defined by a multi-scale spatial grid, as shown in Figure 2.22, and then concatenates the pooled codes to form the final image vector. In this thesis, I conduct a preliminary study on the discriminative optimization of spatial pyramidal pooling for the BoW model (see Chapter 6).

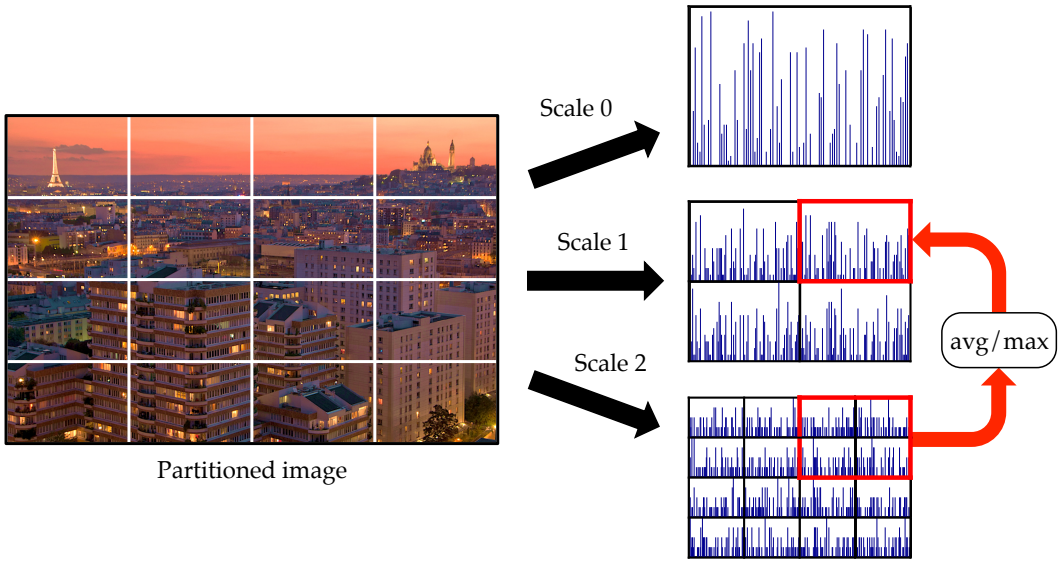


Figure 2.22: Spatial pyramid pooling scheme. A multi-scale grid is used to define spatial partitions to pool the visual codes. Due to the non-overlapping structure of the grids, a grid at a larger scale is an integration – average or max, depending on pooling function used – of its corresponding grids in the smaller scale.

2.4.5 Maximum margin classification

The final step of the BoW model is classification. The objective is to assign a semantic class label to the vectorial representation provided by the pooling step. Numerous techniques have been invented to solve this classification problem through supervised

machine learning. In this thesis, support vector machines (SVM) [Vapnik, 1995] are employed directly as the learning tool for this purpose, but they will not be discussed in detail beyond this section since it is not a focus of this thesis.

The SVM has two main aspects in its design. First, the SVM aims to achieve good separation between classes by finding a set of hyperplanes that maximizes the functional margin, where the distance to the nearest training example is the greatest [Vapnik and Lerner, 1963]. Samples that fall on the margin are known as support vectors. To handle noisy data, a soft-margin method is introduced to allow some examples to violate the margin constraint and be mislabeled [Cortes and Vapnik, 1995]. The second aspect creates nonlinear classifiers by applying the kernel trick [Aizerman et al., 1964] to maximum-margin classifiers [Boser et al., 1992]. This allows the maximum-margin hyperplanes to be fitted in a feature space that has been transformed by a nonlinear kernel function, such as the polynomial function, the Gaussian radial basis function and the hyperbolic tangent function.

In many BoW models that focus on the learning the visual dictionary for image classification, an SVM with a linear kernel is often used [Yang et al., 2009; Perronnin et al., 2010; Zhou et al., 2010]. This is to objectively test the quality of the visual dictionaries learned. Meanwhile, to exploit the full potential of SVMs and boost classification performances, nonlinear SVMs could be employed [Lazebnik et al., 2006; van Gemert et al., 2010; Avila et al., 2013]. Additionally, stochastic gradient descent have been proposed for SVMs to cope with size of very large databases [Bordes et al., 2009; Bordes, 2010].

2.5 Summary and discussion

From this overview, it is clear that machine learning and computer vision are developed in close association. Vision problems serve as a good testbed for new machine learning techniques, while machine learning tools are borrowed to enhance the performances of visual models. The formalism of deep learning methods and the BoW model is now established, allowing for their potential integration.

Learning deep visual representations. One of the main contributions of this thesis is the proposal of a hybrid hierarchical architecture based on RBM-trained visual dictionaries to encode local SIFT descriptors, while exploiting the spatial pyramidal pooling scheme to provide the vectorial representation for image classification. The hybrid architecture merges the complementary strengths of the bag-of-words model and deep architectures. The visual modeling power of local descriptors and pooling

are fused with the adaptability and representational power of deep distributed representations. In the process, various theoretical and applicational aspects of visual coding and pooling are explored.

For shallow unsupervised learning, a new regularization that incorporates desirable coding properties is introduced in Chapter 3. Specifically, the notion of a joint sparse and selective regularization is proposed to enhance the quality of the learned dictionary. Additionally, with deep architectures, it is necessary to implement supervised learning strategies to boost classification performances, which will be investigated in Chapter 4.

The proposed hierarchical architecture achieves the best image classification results among the family of feature coding methods, on both the Caltech-101 [Fei-Fei et al., 2004] and 15-Scene datasets [Lazebnik et al., 2006] (Chapter 5). Experiments demonstrate the importance of supervised fine-tuning for hierarchical visual dictionaries. Inference is also simple and fast as compared to decoder-network-based methods. Qualitatively and quantitatively, the visual representations discovered are concise and capture the underlying structure of image gradients of the SIFT descriptor. Some minor gains have also been observed when attempting to optimize the pooling step (Chapter 6).

3 Regularizing Latent Representations

Chapter abstract — This chapter proposes methods to regularize unsupervised learning based on the knowledge of certain desired structure of the latent representations. The restricted Boltzmann machines will be employed as the basic generative model to establish the proposed regularization algorithm in Section 3.3.1. I will also present several regularization techniques to encourage desirable properties to be incorporated into the latent representation. These include sparsity and selectivity in coding (Section 3.3.2) and transformation invariance (Section 3.3.3). Preliminary studies on the effects of these regularizers are done to understand the effects of the parameterization and presented in Section 3.4.

These methods are based on the unsupervised learning of shallow networks. They form the basis for performing both unsupervised and supervised learning of deep architectures, which will be presented in the next chapter. The methods are also applied to perform visual dictionary learning of local image descriptors, described in Chapter 5.

The material in this chapter has appeared in the following publications:

- Goh, H., Thome, N., and Cord, M. (2010). Biasing restricted Boltzmann machines to manipulate latent selectivity and sparsity. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*. [Goh et al., 2010a]
- Goh, H., Kuśmierz, Ł., Lim, J.-H., Thome, N., and Cord, M. (2011). Learning invariant color features with sparse topographic restricted Boltzmann machines. In *International Conference on Image Processing (ICIP)*. [Goh et al., 2011]

3.1 Introduction

Learning from unlabeled information can help discover interesting properties and the underlying structure of the data. When applied appropriately and effectively, unsupervised learning can complement supervised learning for a data classification task, especially in the absence of sufficient labeled data. One way to improve unsupervised learning for a particular task is to inject additional information, specified by either a priori knowledge or inductive biases [Mitchell, 1980], through the

introduction of regularization. Regularization typically imposes a prior distribution or enforces constraints on a model's parameters during training. A suitable regularization helps a model to prevent overfitting and to tackle ill-posed problems. As a result, having an effective regularization method is almost as important as the statistical learning algorithm itself and they are often employed in unison.

3.2 From neural coding to connectionist models

Our nervous system encodes sensory stimulus by using distributed patterns of electrical neuronal activity. An important property of this coding is that only a subset of neurons are strongly activated in any instance. This notion has been emulated by various connectionist models focusing on unsupervised learning, such as decoder networks [Olshausen and Field, 1996] (see Section 2.2.2) and restricted Boltzmann machines [Lee et al., 2008; Nair and Hinton, 2009] (see Section 2.2.3). In both biological and computational models, there are two important properties of the coding: sparsity and selectivity [Földiák, 2009, 2013].

For human vision and neural network models, neurons representing sensory information, such as visual inputs, have evolved to be sparsely coded to have the following properties:

- increased memory storage capacity [Rolls and Treves, 1990],
- minimized energy and metabolic cost [Barlow, 1989; Levy and Baxter, 1996],
- reduced network connectivity [Boos and Vogel, 1997],
- improved information efficiency [Barlow, 1989; Graham and Willshaw, 1997],
- decreased coding redundancies [Barlow, 1961, 1972], and
- enhanced pattern separability and discrimination [Mitchison and Durbin, 1989; Rolls and Treves, 1990].

Similarly, in computer vision, sparsity in the visual representations also appears to contribute to good performances in image classification [Lee et al., 2009; Yang et al., 2010b].

In this section, I will clearly define sparsity and selectivity (Section 3.2.1) in terms of both neural coding and in our context of learning latent representation. I will also describe a generic method to regularize the learning of restricted Boltzmann machines (Section 3.2.2) and examine the merits and shortfalls of existing selectivity-based regularization methods for RBMs (Section 3.2.3). These discussions form the motivations and set the context for the proposed extension of the regularization algorithm in Section 3.3.1.

3.2.1 What is sparsity and selectivity?

Neural activity can be grouped in either the population or the lifetime domain [Willmore and Tolhurst, 2001; Földiák and Young, 2002]. Sparsity is a property defined in the population domain, while selectivity is characterized in the lifetime domain. The latent representations of a connectionist model can also be described in the same manner. Given a batch of input instances, the latent activations can be defined as a matrix $\mathbf{Z} \in \mathbb{R}^{J \times K}$. Each row $\mathbf{z}_j$ represents the response of a latent variable j with respect to the sequence of K input features, while each column $\mathbf{z}_k$ denotes the latent representation for a given input instance k. The population domain refers to a column $\mathbf{z}_k$ and the lifetime domain refers to a row $\mathbf{z}_j$. Thus, every element z_{jk} in this matrix belongs to one population domain and one lifetime domain, referenced by its corresponding column k and row j indices respectively. These concepts are illustrated in Figure 3.1.

Definitions

Sparsity is a property of a single column in the activation matrix. Given an instance k, the sparsity of the latent representation summarizes the amount of collective activity of the latent vector $\mathbf{z}_k$.

Selectivity is a metric of a row in the activation matrix. The selectivity of a latent variable j is an indication of its activity levels in response to the set of input instances $\mathbf{z}_j$.

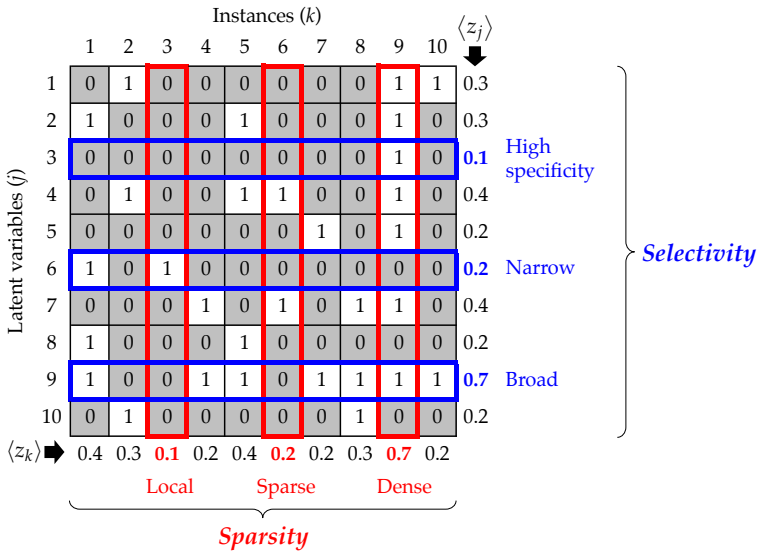
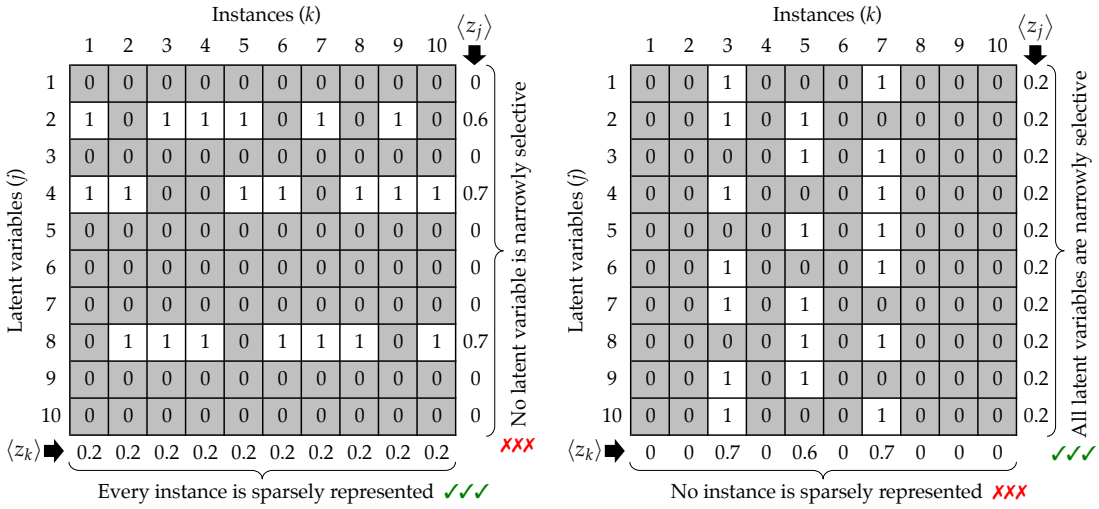


FIGURE 3.1: Definitions and terminology of sparsity and selectivity. The sparsity and selectivity of an activation matrix $\mathbf{Z}$ can be simplistically measured by taking the average activation over the columns and rows respectively. Sparsity is a property of a collection of latent variables in response to an instance, while selectivity is a property of a latent variable across the set of instances. (Terminology adapted from Földiák and Young [2002] and Földiák [2009])

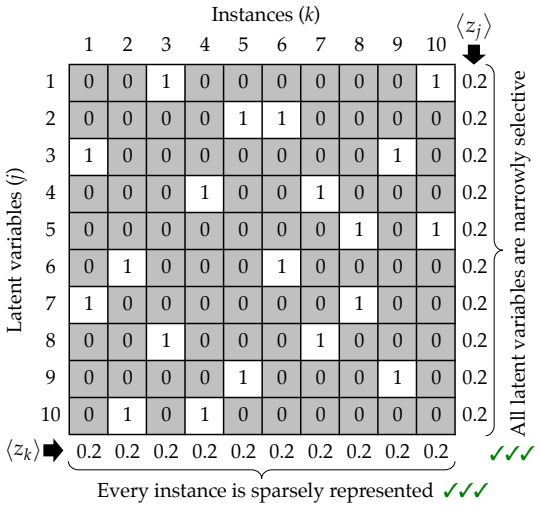
Franco et al. [2007] found that the distributions neuronal recordings of neural activity in both the population and lifetime domains fit positively-skewed long-tail distributions, such as the exponential and gamma distributions. In any given instance, most neurons have low activity and only a few have high responses. Likewise, each neuron fires strongly but rarely in response to a set of input stimulus.

A simple metric for describing the sparsity and selectivity is the mean activation across their corresponding domain. As illustrated in Figure 3.1, just as the level of collective activity of $\mathbf{z}_k$ can range from local, to sparse and dense, the activity $\mathbf{z}_j$ of a latent variable across instances can also range from having high specificity (or grandmother-cell-like) [Konorski, 1967], to being narrow and broad.



(a) Sparsely represented but not narrowly selective.

(b) Narrowly selective but not sparsely represented.



(c) Both sparsely represented and narrowly selective.

Figure 3.2: Importance of balancing sparsity and selectivity. Examples of latent activations z_{jk} show that balancing sparsity and selectivity in the representation gives good differentiation between the instances and diversity among the latent variables.

Using the average activation metric, the average sparsity $\frac{1}{k}\sum_k \langle z_k \rangle$ is mathematically equivalent to the average selectivity $\frac{1}{j}\sum_j \langle z_j \rangle$, which is basically the average activation over the whole matrix. However, beyond having the same average values [Földiák, 2009], sparsity and selectivity are not necessarily correlated [Willmore and Tolhurst, 2001]. This is nowhere more apparent than the toy examples in Figure 3.2. Figure 3.2(a) shows that latent variables need not be narrowly selective even when the latent representation $\mathbf{z}_k$ is sparse for every single instance k. In this case, many latent variables either don't respond to any instance or responds to too many instances. Likewise, as illustrated in Figure 3.2(b), a set of narrowly selective latent variables does not guarantee the existence of sparse population representations. Here, the sparsity for the instances are either too local or too dense. In both cases, the coding strategies do little to improve differentiation between instances, with many latent variables being too similar and redundant in their encoding.

A good coding strategy needs to strike a good balance between sparsity and selectivity. Figure 3.2(c) shows one such example whereby one can observe that different instances have dissimilar latent representations, while each latent variable responds differently to the instances. There is diversity among the latent variables and differentiation between instances. Ideally, the activity of individual latent variables should be mostly silent, while being responsive to a few particular instances and these pattern of responses across the instances should be distinctive from one latent variable to another. Another way to perceive this is that every instance should be represented by only a few latent variables and the representation between instances should not be similar. When this occurs, the variance is low for sparsity across instances and selectivity in the representation. A high variance might indicate an disparity in the representation between instances or latent variables.

3.2.2 Generic restricted Boltzmann machine regularization

A restricted Boltzmann machine (RBM) [Smolensky, 1986] is a bipartite graphical model that approximates the maximum-likelihood of the data distribution using the contrastive divergence algorithm (see Section 2.2.3 for an introduction). It does not explicitly consider the nature of the classification task. Solely using input maximum-likelihood as the only criteria to learn the latent representations might not be the most suitable for the task. How do we embrace a priori knowledge of the desired structure of the latent representations during RBM training? This forms the motivation behind regularizing the learning of the RBM.

The negative-log likelihood loss function of the RBM can be combined with a regularization term $h(\mathbf{z})$ weighted by a constant λ:

$$\mathcal{L}_{RBM+reg} = -\sum_{k=1}^{|\mathcal{D}_{train}|} \log P(\mathbf{x}_k) + \lambda h(\mathbf{z}), \tag{3.1}$$

where $\mathcal{D}_{train}$ refers to the training dataset made up of input-output pairings $\{(\mathbf{x}_k, \mathbf{y}_k)\}$. With regularization, we are able to introduce additional information to control the learning of the latent representation by the RBM. A straightforward way to choose a suitable regularizer $h(\mathbf{z})$ is to define it as a loss function $\mathcal{L}_{reg}$ that is differentiable with respect to the parameters w_{ij}, thus allowing it to directly penalize the learning signal using its gradient $\frac{\partial \mathcal{L}_{reg}}{\partial w_{ij}}$:

$$w_{ij} := w_{ij} + \Delta w_{ij} - \eta \frac{\partial \mathcal{L}_{reg}}{\partial w_{ij}} \tag{3.2}$$

where η is the rate of gradient descent for the regularizer. This modifies the implementation of Algorithm 2.2 to become Algorithm 3.1.

Algorithm 3.1: RBM training with a generic gradient-based regularization

```
Initialize W
repeat
    Get X_0 from randomized training batch
    Sample P_0(Z_0|X_0)                                  // Equation 2.26
    for n = 1 to N do                          // Alternating Gibbs sampling
        Sample P_n(X_n|Z_{n-1})                          // Equation 2.27
        Sample P_n(Z_n|X_n)                              // Equation 2.26
    end
    Update w_ij := w_ij + Δw_ij − η ∂L_reg/∂w_ij         // Equation 3.2
until convergence
```

3.2.3 Low average activation regularization

There are two existing ways to achieve selectivity[1] in RBMs. Lee et al. [2008] proposed to couple the likelihood term with a regularization term (Equation 3.1) that penalizes the latent variables as follows:

$$h(\mathbf{z}) = \sum_{j=1}^{J} \left\| \widetilde{p} - \langle z_j \rangle \right\|_2^2. \tag{3.3}$$

Here, the selectivity of a latent variable is its average activation over the batch of K training instances. The objective parameter $\widetilde{p}$ controls the intended average latent

[1] In the literature, the terms "sparsity" and "selectivity" are sometimes erroneously used interchangeably, sometimes leading to confusion [Földiák, 2009]. We use the term "selectivity" to describe a latent variable's activity across instances, while Lee et al. [2008] defined "sparsity" in the lifetime domain.

activation. Typically, $\widetilde{p}$ is set to be a small constant value to promote low average activation and improve selectivity. An additional penalty term is incorporated into the original update rule (Equation 2.36) based-on the gradient of the regularization term:

$$\Delta w_{ij} = \varepsilon \left(\langle x_i z_j \rangle_{data} - \langle x_i z_j \rangle_{recon} \right) + 2\lambda \left(\widetilde{p} - \langle z_j \rangle_{data} \right) \langle x_i z_j \left(z_j - 1 \right) \rangle_{data}, \tag{3.4}$$

In practice, Lee et al. [2008] does not apply any update to $\mathbf{W}$ and relies on set of latent biases w_{0j} alone to control the activation of the latent variables. However, Hinton [2010] remarked that this may result in highly negative biases to induce the required degree of selectivity, which is compensated by highly positive weights to ensure the latent variables remain useful.

Another drawback of this method is that as z_{jk} nears zero or one, the derivative of the sigmoid function approaches zero, thus diminishing any opposing learning signal. This may not be desirable, especially if z_{jk} is high but should be low. Another further simplification is performed in practice to alleviate this problem, such that the update rule is modified to be:

$$\Delta w_{0j} = \varepsilon \left(\langle z_j \rangle_{data} - \langle z_j \rangle_{recon} \right) + 2\lambda \left(\widetilde{p} - \langle z_j \rangle_{data} \right), \tag{3.5}$$

as reported by Ekanadham [2007], while the weight updates remain as those defined by contrastive divergence Equation 2.36. Appendix A.2 details the process of this simplification. This simplification bares similarity to the cross-entropy penalty between the observed average activation and desired average activation used by Nair and Hinton [2009]:

$$h(\mathbf{z}) = \sum_{j=1}^{J} -\widetilde{p} \log q_{j,t} - (1 - \widetilde{p}) \log \left(1 - q_{j,t} \right). \tag{3.6}$$

where $q_{j,t}$ is a more complex "exponentially decaying average of the mean probability" used to characterize the selectivity of a latent unit, defined as [see Hinton, 2010]:

$$q_{j,t} = (1 - \tau) \langle z_j \rangle_t + \tau q_{j,t-1}, \tag{3.7}$$

where t denotes the current mini-batch used for training and τ controls the rate of decay. $\mathbf{W}$ is updated with the gradient of Equation 3.6, which is claimed by Nair and Hinton [2009] and Hinton [2010] to be simply $q_{j,t} - \widetilde{p}$ for logistic units. This simplified penalty term is only possible if some assumptions are made when calculating the derivative (see Appendix A.3 for additional discussions).

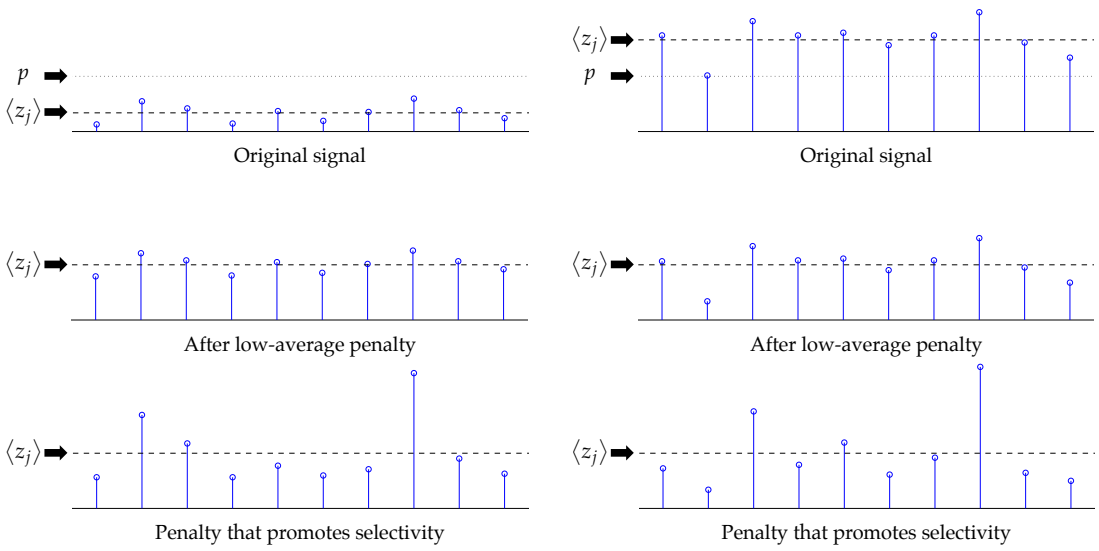
Implementing these regularization methods is simple - perhaps even to the extent of being over-simplified. So, they are susceptible to the following drawbacks.

- *Low average activation may not be selective.* For a latent variable to be selective, it should respond strongly to only a few instances and non-responsive to others. However, these methods merely attempt to get the average activation to be low. Even when the selectivity objective p is satisfied, the latent variable may not be selective. An example of this occurs when $z_{jk} = \widetilde{p}, \forall k \in K$.

- *Penalty does not explicitly promote selectivity.* The penalty, based on a single average statistic, modifies the latent activations equally across the instances. This is counter-intuitive to the concept of selectivity, where a variety of high and low activations may be preferred. When $\langle z_j \rangle < \widetilde{p}$, a more selective activation sequence can be obtained by discriminately increasing the higher activations, while keeping the rest low (see Figure 3.3(a)). Similarly, when $\langle z_j \rangle > \widetilde{p}$, the smallest activations could have the priority to be lowered first (see Figure 3.3(b)).

- *Sparsity is not guaranteed.* The methods regularize for only selectivity in the lifetime domain and do not consider sparsity in the population domain. This leaves them potentially open to the drawbacks of a non-balanced coding scheme as explained in Section 3.2.1 and illustrated in Figure 3.2(b), such as having indistinguishable latent variables or overly broad or local coding.

Section 3.3 proposes a method to overcome the limitations above.



(a) When $\langle z_j \rangle < \widetilde{p}$. (b) When $\langle z_j \rangle > \widetilde{p}$.

Figure 3.3: Explicitly promoting selectivity. For both cases, where (a) $\langle z_j \rangle < \widetilde{p}$ (left) or (b) $\langle z_j \rangle > \widetilde{p}$ (right), an average-based penalty gets $\langle z_j \rangle$ to match p by shifting all activations equally but does not explicitly promote selectivity. However, the same $\langle z_j \rangle$ can be obtained by increasing the higher activations and decreasing the lower activations, leading to a more selective activation pattern. Although the final $\langle z_j \rangle$ is the same, their activation distributions are very different.

3.2.4 Sparsifying logistic

A selective activation sequence can be generated by the sparsifying logistic function [Ranzato et al., 2006]. The function implicitly regularizes the parameters when training a decoder network (Section 2.2.2). Although the regularization is different, from the methods described for RBMs above, it is useful to study it because it generates spike-like activations that deals with the limitations of the low-average-based regularization. The function transforms the activation of each latent variable based on an exponentially decaying weighted sum of historic activation values:

$$\hat{z}_{jk} = \frac{\eta \exp(z_{jk})}{\zeta_j(k)} \qquad \text{with} \qquad \zeta_j(k) = \eta \exp(\beta z_{jk}) + (1-\eta)\zeta_j(k-1), \tag{3.8}$$

where $\eta \in [0, 1]$ controls the level of selectivity by determining the number of instances to sum over, while β controls the softness of this softmax function. For large values of β and small values of η, brief punctuated activations in time is generated.

However, due to the exponential temporal weighting, the activation is not independent of the activation sequence. The same set of activations presented in a different sequence may produce different transformed outputs for the same activation level. As a result, the ordinality of the activation levels in the set is not preserved.

3.3 Representation regularization and regularizer design

In this section, a new regularization function is proposed, which is formulated such that there will be more control over the regularization process (Section 3.3.1). Furthermore, the function is generic enough to allow for different types of regularizers to influence the learning of the latent representations, which will be described from Sections 3.3.2 and 3.3.3.

3.3.1 Point- and instant-wise regularization

Consider an activation matrix $\mathbf{Z} \in \mathbb{R}^{J \times K}$, with each row $\mathbf{z}_j$ representing the response of a latent variable with respect to the set of K input features, while each column $\mathbf{z}_k$ denoting the latent representation for a given input instance k. To gain a more precise control when manipulating the learning of the latent representations is to bias the activations for each latent variable j (point-wise) given each training example k (instance-wise). Biasing each point-instance configuration z_{jk} gives control over individual elements in the activation matrix $\mathbf{R}$. This can be achieved by specifying a target activation matrix $\mathbf{P} \in \mathbb{R}^{J \times K}$, where $p_{jk} \in [0, 1]$ for a binary RBM.

A regularizer $h(\mathbf{z})$ from Equation 3.1 can be defined using the cross-entropy loss, summed over J latent variables and the dataset, which equates to

$$\begin{aligned} h(\mathbf{z}) &= -\sum_{k=1}^{|\mathcal{D}_{train}|}\sum_{j=1}^{J}\log P(p_{jk}|z_{jk}) \\ &= -\sum_{k=1}^{|\mathcal{D}_{train}|}\sum_{j=1}^{J} p_{jk}\log z_{jk} + (1-p_{jk})\log(1-z_{jk}), \end{aligned} \tag{3.9}$$

for binary latent variables. Minimizing the cross-entropy between the $\mathbf{P}$ and $\mathbf{Z}$ penalizes activations that are different from the target activations. Merging the gradient of the cross-entropy loss with the original RBM contrastive divergence learning rule (Equation 2.36), the regularized parameter updates can be written as follows:

$$\Delta w_{ij} = \varepsilon\left(\langle x_i s_j\rangle_{data} - \langle x_i z_j\rangle_{recon}\right), \tag{3.10}$$

where

$$s_{jk} = (1-\phi)\, z_{jk} + \phi p_{jk}, \tag{3.11}$$

is the modified activation matrix used to update the parameters. It is defined as an element-wise weighted sum between a latent activation z_{jk} and its corresponding target activation p_{jk}. Please refer to Appendix A.1 for the derivation of the new RBM learning rule with regularization.

Meanwhile, alternating Gibbs sampling is still performed between the two layers to estimate the equilibrium distribution of the model. As illustrated in Figure 3.4, since $\mathbf{Z}$ and $\mathbf{S}$ are the same size, the regularization retains the form of the original contrastive divergence update rule (Equation 2.36). This regularization is extremely versatile, as it is able to penalize the RBM learning based on any target representation. In this sense, the job of regularizing the representation is shifted to the design of suitable targets $\mathbf{P}$.

Here, ϕ is a hyperparameter. The influences of p_{jk} and z_{jk} are regulated by ϕ. If ϕ is constrained to be between 0 and 1, then $s_{jk} \in [0,1]$ can be seen as the revised activation probability of z_{jk}. When $\phi = 0$ or if the target activation is matched (i.e. $z_{jk} = p_{jk}$), then the parameter updates simplify to those of the original contrastive divergence learning algorithm.

From the new update rule (Equation 3.10), a new optimization algorithm (Algorithm 3.2) can be developed. The regularization manifests in four steps. First, alternating Gibbs sampling is performed like in a conventional RBM (Lines 5 to 8). The target $\mathbf{P}$ is then obtained (Line 9) and combined with $\mathbf{Z}_0$ to form $\mathbf{S}$ (Line 10), which in turn is used to update the RBM's parameters (Line 11). This process is iterated until a stopping criterion is met.

By using targets **P** to update the RBM's parameters **W**, the algorithm is perfectly suited for alternating optimizations between **P** and **W**, if required. This is similar to the training algorithm for decoder networks (Section 2.2.2), where an alternating algorithm is useful when there are dependencies between the representation and parameters and both needs to be optimized.

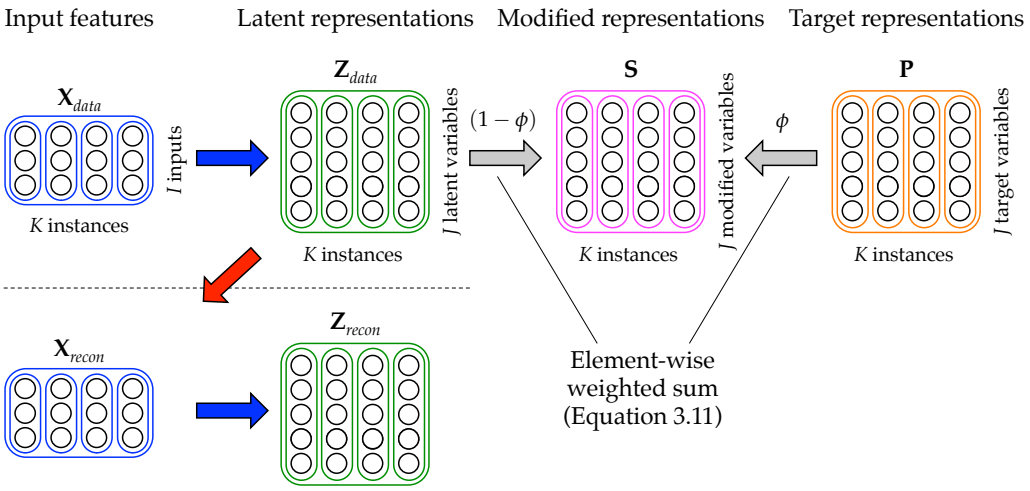


FIGURE 3.4: Point- and instant-wise regularization. The latent activation matrix $\mathbf{z}_{data}$ is merged with the target matrix **P** via a weighted sum modulated by a hyperparameter ϕ to produce a modified activation matrix **S**, which will be used for parameter updates. Meanwhile, alternating Gibbs sampling is still performed from $\mathbf{Z}_{data}$ to estimate the equilibrium distribution.

Algorithm 3.2: RBM training with point- and instance-wise regularization

Initialize **W**
repeat
  Get $\mathbf{X}_0$ from randomized training batch
  Sample $P_0(\mathbf{Z}_0|\mathbf{X}_0)$ // Equation 2.26
  for $n = 1$ *to* N **do** // Alternating Gibbs sampling
    Sample $P_n(\mathbf{X}_n|\mathbf{Z}_{n-1})$ // Equation 2.27
    Sample $P_n(\mathbf{Z}_n|\mathbf{X}_n)$ // Equation 2.26
  end
  Get **P** // See Sections 3.3.2 and3.3.3
  Compute $\mathbf{S} = (1-\phi)\mathbf{Z}_0 + \phi\mathbf{P}$ // Equation 3.11
  Update $w_{ij} := w_{ij} + \varepsilon(\langle x_i s_j \rangle_{data} - \langle x_i z_j \rangle_{recon})$ // Equation 3.10
until *convergence*

3.3.2 Generating jointly sparse and selective representations

The motivation of balancing sparsity and selectivity in the latent activity matrix has been explained in Section 3.2.1. Enforcing sparsity promotes instance-wise competition and lateral inhibition between the latent variables. Imposing selectivity avoids over-dominance of the instances by any individual latent variables, while avoiding

having silent ones. Meanwhile, it is worthwhile to extend the existing selectivity regularization methods (Section 3.2.3) to overcome their drawbacks. In this section, I propose a method to construct a target activation matrix $\mathbf{P} \in \mathbb{R}^{J \times K}$ as inputs to the point- and instance-wise regularizer described in Section 3.3.1.

As previously mentioned in Section 3.2.1, neural activity in both the population and lifetime domains fit positively-skewed long-tail distributions, such as the exponential and gamma distributions. In our context, the target matrix $\mathbf{P}$ can be designed by fitting the latent activations to such distributions in both the columns and the rows to achieve sparsity and selectivity. As a result, in each domain, there will be some latent activations that will be highly activated, while most will be silent.

Selectivity in the rows. For a latent variable to achieve selectivity, the K-dimensional data for each row of activation probabilities $\mathbf{z}_j$ is transformed via a two step procedure (Figure 3.5).

Step 1 transforms the original activation sequence (Figure 3.5(a)) into a normalized uniform distribution (Figure 3.5(b)). The fractional rank for each element relative to its row is calculated. The smallest element is mapped to 1, the largest the value of K and the mean of the ordinal ranking is assigned in the event of a tied. The values are scaled to the interval of $[0, 1]$, where the smallest and largest values are assigned $\frac{1}{2K}$ and $\frac{2K-1}{2K}$ respectively. The data transformation for the vector $\mathbf{z}_j$ can be written as follows:

$$\hat{\mathbf{z}}_j = norm\_rank(\mathbf{z}_j) = \frac{1}{K}(rank(\mathbf{z}_j) - 0.5), \tag{3.12}$$

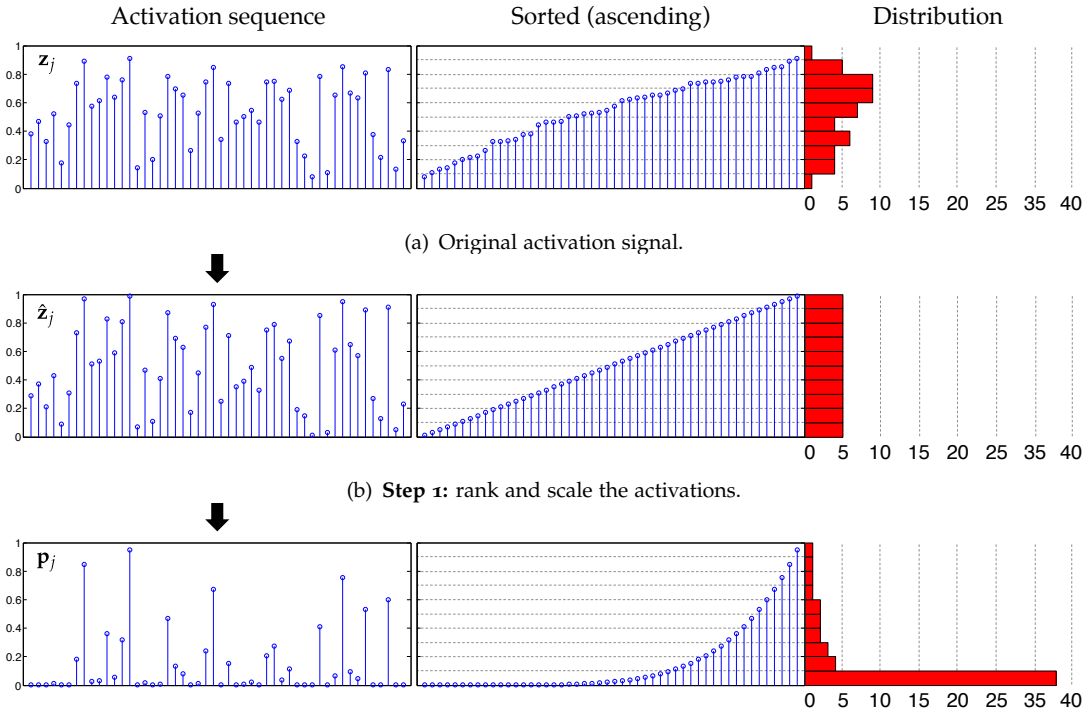
where $rank(\cdot) \sim \mathcal{U}(1, K)$ is computed for each element in the vector and normalized across the entire vector, resulting in $\hat{z}_j \sim \mathcal{U}(0, 1)$.

Step 2 maps the resulting activations of step 1 to a positively-skewed long-tail distribution (Figure 3.5(c)). The following power law distribution is chosen for this mapping to keep the result in the original interval:

$$\mathbf{p}_j = \hat{\mathbf{z}}_j^{\gamma} = \left(norm\_rank(\mathbf{z}_j)\right)^{\gamma}, \tag{3.13}$$

where γ is used to create the power law expression and control the desired selectivity level, and $\hat{\mathbf{z}}_j = norm\_rank(\mathbf{z}_j)$ is the vectorial result of step 1. A more intuitive parameter to express the level of selectivity is to use the target mean activation of a latent variable

$$\mu = \int_0^1 \left(\hat{z}_j \sim \mathcal{U}(0, 1)\right)^{\gamma} d\hat{z}_j = \frac{1}{\gamma + 1}. \tag{3.14}$$



(a) Original activation signal.

(b) **Step 1:** rank and scale the activations.

(c) **Step 2:** transform the ranked signal to fit a long-tailed distribution.

FIGURE 3.5: Transforming a sequence of latent activations to their targets. A succession of two transformation steps is performed to obtain the target activation $\mathbf{p}_j$ from $\mathbf{z}_j$. For illustrative purposes, the sequence of latent activations on the left is sorted in an ascending order of their activation level (middle) and its histogram is displayed on the right. (a) The original activation sequence may take the form of any empirical distribution. (b) Step 1 ranks the signals and scales them between 0 and 1, resulting in a uniform distribution within that interval. (c) Step 2 maps the ranked signals to fit a predefined long-tailed distribution to obtain $\mathbf{p}_j$.

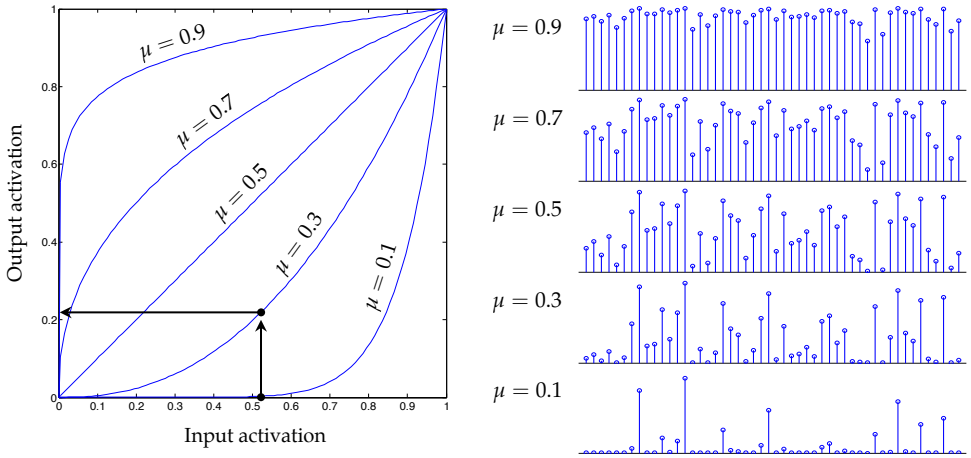


FIGURE 3.6: Activation mapping curves and their activation sequences. (Left) Curves for power law distributions that map an input activation to an output activation. The different curves are obtained by varying the target mean activation μ in the power law expression. (Right) An example of how the form of an activation sequence varies with μ.

This updates the data transformation equation to:

$$\mathbf{p}_j = \left(norm\_rank(\mathbf{z}_j)\right)^{\frac{1}{\mu}-1}, \tag{3.15}$$

where $\mu \in]0,1[$. If $\mu < 0.5$, the uniform distribution is mapped to one that is positively skewed and the code is more narrowly selective. As μ is reduced, the activation strength and frequency are decreased. Figure 3.6 shows the activation mapping curves and activation behavior with varying values of μ. From a coding perspective, μ dictates the amount of feature sharing and differentiation between instances.

This two-step operation leads to a direct solution for selectivity without the need for any heavy optimization step. Those familiar with image processing, will find it similar to performing histogram equalization followed by a gamma correction, with a expansive gamma for increased selectivity.

Instead of merely getting the RBM to have low average activations (Section 3.2.3), the activations are selectively and individual biased such that each row collectively forms a positively skewed distribution, with only a few highly activated responses. Unlike the sparsifying logistic (Section 3.2.4), the proposed data transformation is independent of the temporal order of instances within the training batch and preserve the ordinal ranking of the activation levels in each row.

Sparsity in the columns. The columns of matrix $\mathbf{Z}$ can be sparsified in the same way the rows were made selective. A set of J activations in each column could be transformed with a similar formulation as Equation 3.15:

$$\mathbf{p}_k = \left(norm\_rank(\mathbf{z}_k)\right)^{\frac{1}{\mu}-1}. \tag{3.16}$$

Now, μ_s is the intended mean activation of the set of J latent variables in response to a given instance k. From a coding perspective, μ controls the amount of similarity and differentiation between the latent variables in the population.

Combining selectivity and sparsity. According to Willmore and Tolhurst [2001], sparsity and selectivity are not highly correlated. This means that achieving selectivity does not guarantee that the representations are sparse, and vice versa (see Figure 3.2 and explanation in Section 3.2.1). Hence, there is a need to combine both properties into the target matrix $\mathbf{P}$.

I found that neither transforming z_{jk} based on its relative rank in the matrix $\mathbf{Z}$, nor taking the Hadamard product of the objective matrices $\mathbf{P}_{sparse} \circ \mathbf{P}_{selective}$, were effective in producing a desirable target matrix. Instead, repeatedly performing column-wise

sparsification (Equation 3.16) followed by inducing row-wise selectivity Equation 3.15, for a number of repetitions R, yields a target matrix $\mathbf{P}$ that is both sparse and selective. In practice, due to the iterative nature of RBM training, one alternating pass was performed for each iteration ($R = 1$). The algorithm for regularizing the RBM integrated with the sparse and selective data transformation is described in Algorithm 3.3.

Unlike the other previous methods [Lee et al., 2008; Nair and Hinton, 2009], both selectivity and sparsity can now be induced by when training RBMs. By inducing both selectivity and sparsity, the regularizer attempts to more explicitly associate specific latent variables to specific instances till done in a purely unsupervised manner.

Algorithm 3.3: RBM with jointly sparse and selective regularization

Initialize $\mathbf{W}$
repeat
  Get $\mathbf{X}_0$ from randomized training batch
  Sample $P_0(\mathbf{Z}_0|\mathbf{X}_0)$ // Equation 2.26
  for $n = 1$ *to* N **do** // Alternating Gibbs sampling
    Sample $P_n(\mathbf{X}_n|\mathbf{Z}_{n-1})$ // Equation 2.27
    Sample $P_n(\mathbf{Z}_n|\mathbf{X}_n)$ // Equation 2.26
  end
  for $r = 1$ *to* R **do** // Sparse and selective data transformation
    for $k = 1$ *to* K **do** // Sparsify instances
      Compute $\hat{\mathbf{p}}_k = (norm\_rank(\mathbf{z}_k))^{\frac{1}{\mu}-1}$ // Equation 3.16
    end
    for $j = 1$ *to* J **do** // Selectivity transform on latent variables
      Compute $\mathbf{p}_j = \left(norm\_rank(\hat{\mathbf{p}}_j)\right)^{\frac{1}{\mu}-1}$ // Equation 3.15
    end
  end
  Compute $\mathbf{S} = (1-\phi)\mathbf{Z}_0 + \phi\mathbf{P}$ // Equation 3.11
  Update $w_{ij} := w_{ij} + \varepsilon(\langle x_i s_j\rangle_{data} - \langle x_i z_j\rangle_{recon})$ // Equation 3.10
until *convergence*

3.3.3 Inducing topographic organization

In the previous section, I proposed a method to use the point- and instance-wise regularization (Section 3.2) to achieve jointly sparse and selective latent representations (Section 3.3.2) for the purpose of improving representational discrimination between instances and latent variables. This section demonstrates the generality of the regularization method by extracting the intrinsic similarity between the latent variables from the input data. This is done by organizing the latent variables into a 2D lattice, known as a topographic map. The objective of topographic organization is to learn representations that exhibit low-level invariance to various image transformations, such as translation, rotation and scaling, and other distortions, such as illumination variations.

A two-layered network to model the sparsity and topological organization of simple and complex cells was introduced by Hyvärinen and Hoyer [2001]. In their model, the connections from the inputs to the first layer are learned weights, while the connections between the first and second layers are fixed such that energies of units in layer one are locally pooled. The output of layer two is maximized for sparseness and then used as a signal to learn the weights between the input layer and layer one.

A similar two-layered scheme is adopted as shown in Figure 3.7. Latent variables $\mathbf{z}_k$ are first activated by an input image patch $\mathbf{x}_k$. The activations are organized into a 2D feature map and a new set of spatially pooled activations $\hat{\mathbf{p}}_k$ is computed. Subsequently, selectivity and sparsity, as described in Section 3.3.2, are introduced to obtain targets $\mathbf{P}$ from $\hat{\mathbf{P}}$, which is used to regularize updating of parameters for the RBM using the point- and instance-wise regularization (Section 3.2).

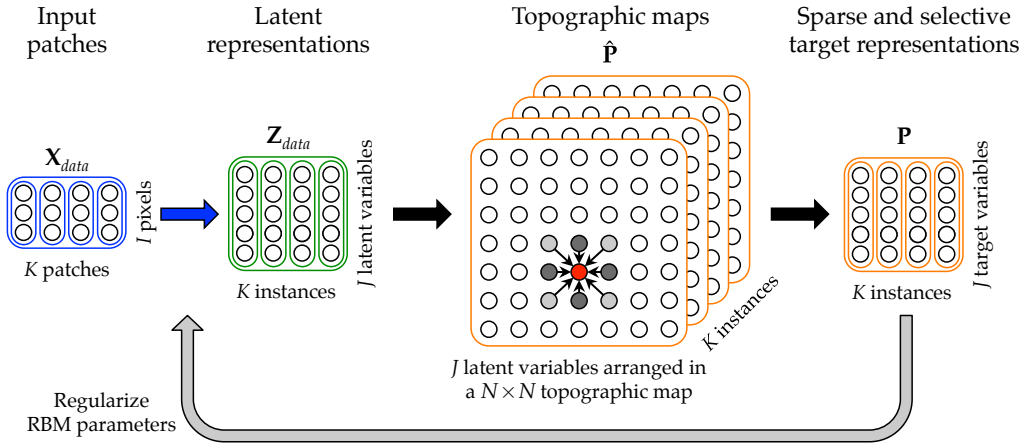


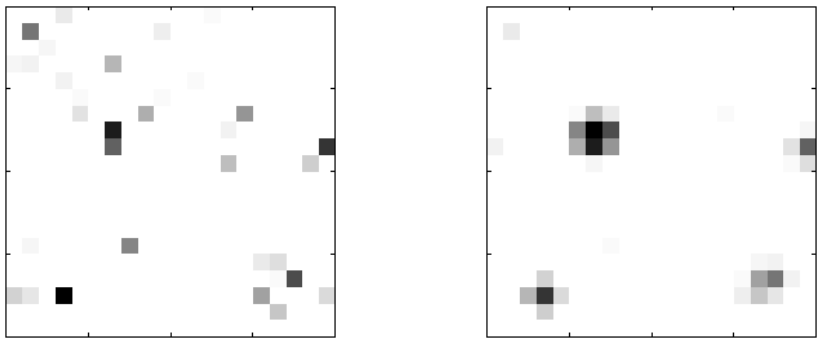
Figure 3.7: Framework for inducing topographical organization. From a batch of K image patch inputs $\mathbf{X}_{data}$, the latent variables are activated $\mathbf{Z}_{data}$ via learned weights. The latent activations are organized into a 2D topographic feature map $\hat{\mathbf{P}}$ and locally pooled via fixed weights. Subsequently, selectivity and sparsity are induced to obtain $\mathbf{P}$. Finally, $\mathbf{P}$ is used to regularize the learning of the RBM parameters using a point- and instance-wise regularizer.

As shown in Figure 3.7, the latent activations are organized into a $N \times N$ feature map $\hat{\mathbf{P}}$. A topographic organization is structured by introducing spatial dependencies between neighboring latent variables. Each $\hat{p}_{jk}$ pools activations from its local neighborhood. This has the same effect as convolving a K-dimensional local filter over the stack of K topographical maps. One way to implement this is to enforce the topography through a fixed set of pooling weights $\omega(j, \hat{\jmath})$ between two latent variables j and $\hat{\jmath}$. The topographic activations $\hat{p}_{jk}$ are computed as

$$\hat{p}_{jk} = \sum_{\hat{\jmath}=1}^{J} z_{\hat{\jmath}k} \omega(j, \hat{\jmath}) \tag{3.17}$$

where $\omega(j, \hat{\jmath})$ is a neighborhood function based on the topographic distance between two latent variables j and $\hat{\jmath}$. A Gaussian kernel with wrap around was used for this

work. Figure 3.8 shows an example of a resulting topographical map as compared to an independently coded one without spatial dependencies.



(a) Independent coding (b) Topographic coding

FIGURE 3.8: Comparing independent and topographic coding. Each grid shows the activation of a latent variable in the feature map, where a darker color denotes a higher activation. (a) Independent coded activations of the original model have no spatial structure. (b) After topographic organization is introduced, the latent activations are mutually supportive and spatially clustered.

3.4 Representation regularization experiments

This section describes experiments performed and provides both qualitative and quantitative analysis on regularizing latent representations. The objectives of the experiments are to demonstrate the transfer of representational properties from the target matrix to the model, provide an understanding of the effects of model parameterization and reinforce the importance of the various regularizers.

3.4.1 Visualization: modeling natural image patches

Through the latent layer, RBMs are able to model the interactions between input dimensions. As a result, they are suitable for extracting the underlying structure in images. The result of the learning can be visualized to have an understanding of the structure of the parameters learned.

Experimental dataset and setup. An RBM, regularized with sparse and selective targets, is trained to model the statistics of natural image patches from the Kyoto natural images dataset [Doi et al., 2003][2]. The dataset contains a set of 62 calibrated natural images corresponding cone photoreceptor responses separated into three components: long-cone, medium-cone and short-cone. As recommended by the authors of the dataset, only the long components were used to generate grayscale samples.

[2] http://www.cnbc.cmu.edu/cplab/data_kyoto.html

To form the training set, 100,000 patches of size 14×14 pixels were randomly sampled from the 62 images in the dataset. The RBM disregards spatial layout of the input dimensions, even when the inputs are spatially consistent image patches. Instead, it tries to learn to discover this structure automatically through learning. The image patches (14×14 pixels) were reshaped to a single vector of size 196. The RBM with 196 input dimensions and 400 latent variables was initialized and trained. Each cycle through the entire dataset is denoted as an epoch. A total of 100 epochs were performed to learn the weight parameters of the network.

Visualization of weight parameters. The vector of parameters $\mathbf{w}_j \in \mathbb{R}^{196}$ used to project the input to the latent variable z_j can be visualized as a filter in the original image space. Reversing the process in which the input vector is reshaped from the image patch, the filter can be formed by reshaping the 196-dimensional weight vector $\mathbf{w}_j$ into a 14×14 structure. This can be perceived as filter over the entire input image space, as illustrated in Figure 3.9. When this is done for all J latent variables, then the collection of filters is known as a filter bank.

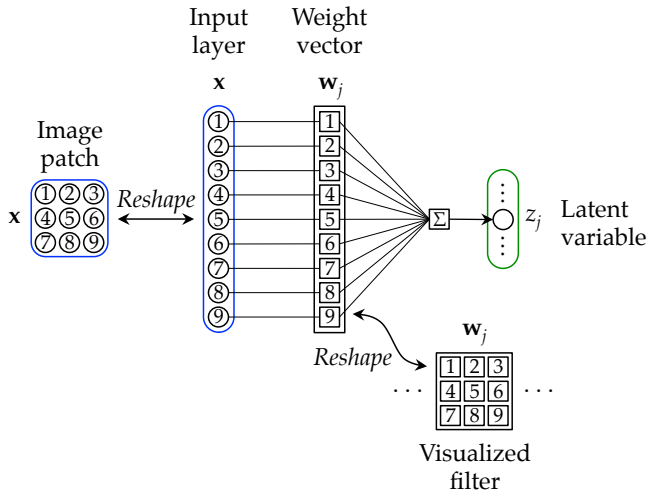
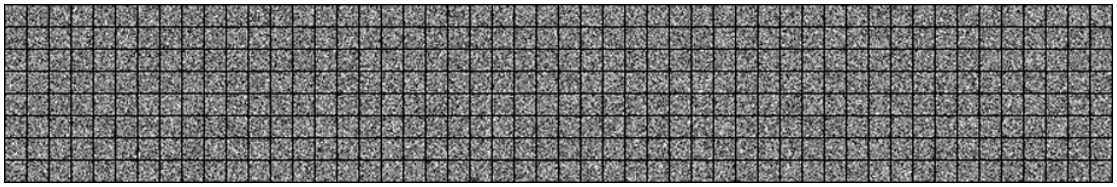
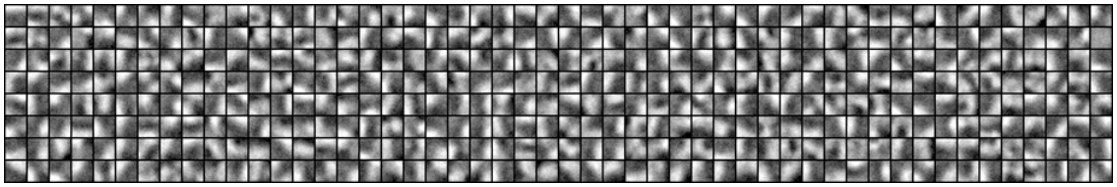


Figure 3.9: Visualization of weight parameters as a filter. The weight parameters $\mathbf{w}_j$ corresponding to a latent variable z_j can be projected back and reshaped to the input image space and visualized as a filter.

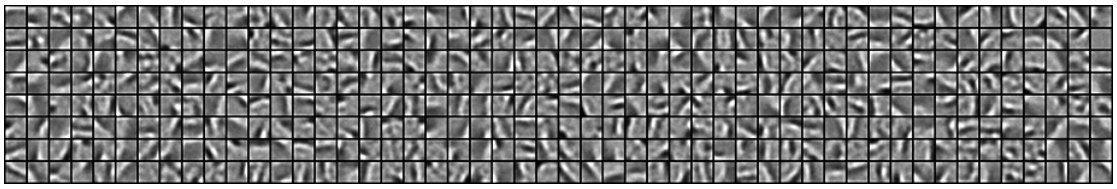
A filter bank of all 400 filters was visualized. Figure 3.10 shows the evolution of the filters during training. Figure 3.10(a) shows the initial random parameters for the RBM. At the end of the training, a bank of Gabor-like filters, as shown in Figure 3.10(d), were learned from the natural image patches. The filters exhibit Garbor-like tuning with a diverse combination of appearances in terms of orientation, spatial position and spatial frequency. This result is visually consistent with other related methods [Ranzato et al., 2006; Lee et al., 2008; Doi and Lewicki, 2005; Teh et al., 2004; Olshausen and Field, 1996].



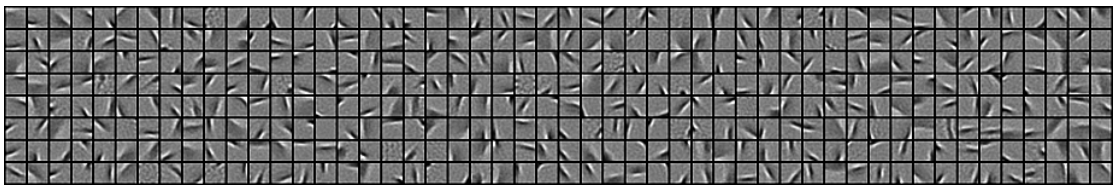
(a) Initial filters (before training).



(b) At epoch 1.



(c) At epoch 10.



(d) At epoch 100.

FIGURE 3.10: Filter bank of a sparse and selective RBM. Each square represents a visualized filter of the weight vector corresponding to a latent variable. The appearance of the filters evolve during the process of learning from natural image patches, from (a) initialization through to the (d) final epoch.

3.4.2 Experiment: modeling handwritten digit images

Experimental dataset. The MNIST handwritten digit dataset [LeCun et al., 1998] was used to conduct studies on the effects of parameterization on an image classification task with sparse and selective regularizers. The dataset contains images of handwritten digits of 28×28 pixels each (Figure 3.11). It is split into $60,000$ training and $10,000$ test images. Each image belongs to one of ten classes, corresponding to a digit: $\{$"0", "1", "2", "3", "4", "5", "6", "7", "8", "9"$\}$.

FIGURE 3.11: MNIST handwritten digit dataset. The dataset consists of handwritten digit images, each belonging to one of ten classes.

General setup and objectives. In the experiments, RBMs with 784 input dimensions and 1,000 latent variables were used to model the images. When trained using an RBM, which had been jointly regularized with sparsity and selectivity using a suitable parameterization, the learned filters appear to encode localized handwritten strokes, as shown in Figure 3.12.

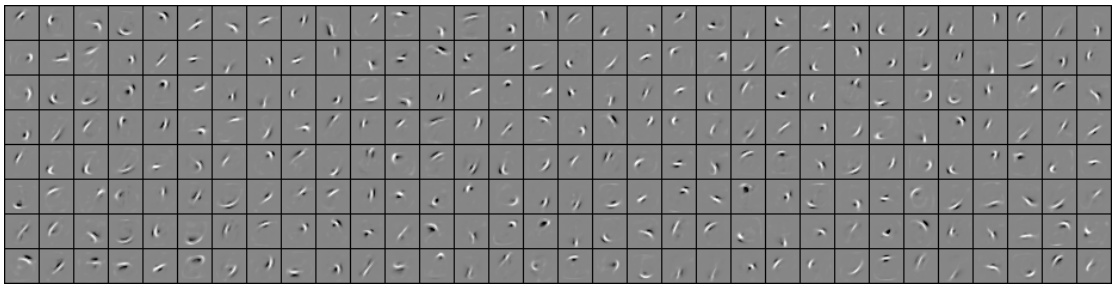


Figure 3.12: Examples of filters learned by an RBM regularized with sparse and selective targets, when trained on handwritten digit images.

The desired mean latent activation μ was the main parameter varied and its effects were analyzed. With a varying μ, two studies with the following objectives were conducted.

1. *Regularizer induction.* Verify that selectivity and/or sparsity in the target matrix have been transferred to the representations encoded by the learned parameters.

2. *Representational discrimination.* Understand the relationship between the level of selectivity and sparsity induced and the amount of class-based discrimination in a supervised task.

In both studies, RBM training was performed on the training set, while the test set was used for empirical analysis.

Regularizer induction. This study seeks to verify that the representational properties of selectivity and sparsity from the target matrix $\mathbf{P}$ are being transferred to the RBM parameters during learning. This study also aims to show the importance of having a jointly sparse and selective regularizer over a regularizer that has only one of those properties.

Evaluation metric. RBMs with different values of μ between 0 and 1 in regular intervals of 0.1 were trained. The boundary values of 0 and 1 were replaced by 0.001 and 0.999, since μ cannot be set as 0 or 1. The amount of induced sparsity and selectivity were measured for the different parameterization settings. The amount of selectivity and sparsity for a set of activations were measured using the activity ratio [Treves and Rolls, 1991], which accounts for the length of the tail of the distribution

of activations. Given an activation row $\mathbf{z}_j \in \mathbb{R}^K$, its selectivity can be described by the activity ratio

$$a_j = \frac{\left(\frac{1}{K}\sum_{k=1}^{K} z_{jk}\right)^2}{\frac{1}{K}\sum_{k=1}^{N} z_{jk}^2}, \tag{3.18}$$

where $a_j \in [0, 1]$. Similarly, an activation column $\mathbf{z}_k \in \mathbb{R}^J$ can be measured as

$$a_k = \frac{\left(\frac{1}{J}\sum_{j=1}^{J} z_{jk}\right)^2}{\frac{1}{J}\sum_{j=1}^{J} z_{jk}^2}. \tag{3.19}$$

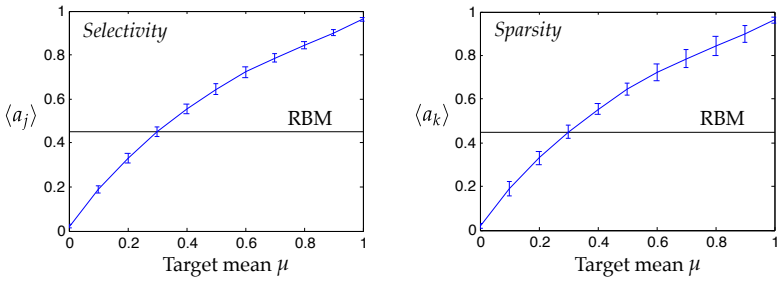
An activity ratio nearer to 0 indicates either a representation with a more local sparsity or a higher specificity. The mean and variance of a_j across the $1,000$ latent variables and mean and variance of a_k over the $10,000$, were recorded as $\langle a_j \rangle$ and $\langle a_k \rangle$ respectively.

Evaluation results and discussion. Using input data from the MNIST test set, the activity ratios for both the rows and the columns of activation matrix $\mathbf{Z} \in \mathbf{R}^{1,000 \times 10,000}$ were measured as the target μ was varied. As shown in Figure 3.13(a), as μ decreases, both activity ratios mimic the decrease. This monotonic relationship implies that the properties of selectivity and sparsity have been transferred from the target activations to the parameters.

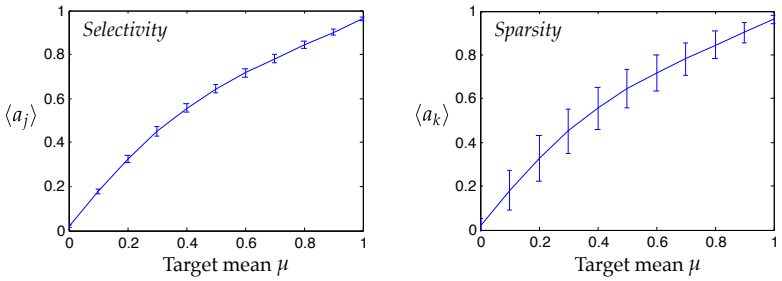
Selective but not sparse. Regularizing the RBM with only selectivity results in activity ratios with a large spread in the population domain (Figure 3.13(b)). With a large activity ratio spread for sparsity, there will be more instances that are locally or densely coded rather than sparsely represented. Explicitly increasing the selectivity improves differentiation between the instances, but not necessarily make the latent variables more distinguishable from each other.

Sparse but not selective. Conversely, regularizing only for sparsity results in a high activity ratio variance in the lifetime domain (Figure 3.13(c)). This means that the latent variables may have high specificity or broad tuning. Given a set of input instances, there may be some latent variables that responds too often or are too silent. Controlling the sparsity does not automatically result in the control of the tuning of the selectivity.

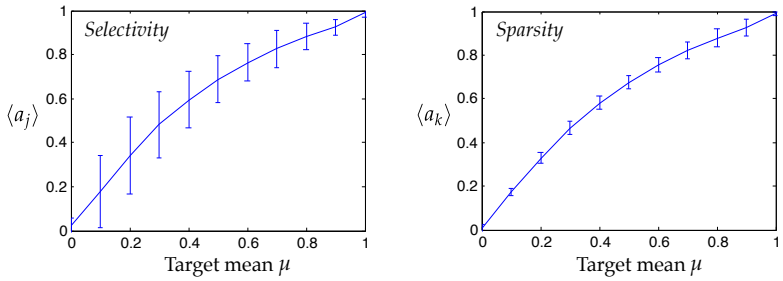
Jointly sparse and selective. A jointly sparse and selective regularizer reduces the number of overly active or silent units, and superfluously or inadequately represented instances. This is evident in Figure 3.13(a), whereby the variance of activity ratios are low for both sparsity and selectivity. Please refer to Section 3.2.1 for a theoretical discussion of this phenomenon.



(a) Regularized jointly with sparsity and selectivity.



(b) Regularized with only selectivity.



(c) Regularized with only sparsity.

Figure 3.13: Analysis of activity ratios with respect to the target mean μ. (a) Both sparsity and selectivity decrease in relation to μ, showing a transfer in representation from the target to the actual latent activations. As a reference, the activity ratios for the unregularized RBM are plotted. (b) By regularizing only for selectivity, the spread of activation ratios in the population domain is high. (c) Regularizing only for sparsity results in high variations in the lifetime domain. (a) In contrast, the jointly regularized representations have low variances in activity ratios.

Representational discrimination. Does the level of sparsity and selectivity induced in the representation improve discrimination for a supervised task? This is the main question that will be empirically evaluated in this study. This can again be done by varying μ. The study was conducted in the range of $0.001 \leq \mu \leq 0.12$.

Evaluation metrics. Two metrics were used to understand the relationship between the amount of selectivity and sparsity induced and the amount of class-based discrimination in a supervised task. Let $\mathbf{y} \in \mathbb{R}^C$ be the vector of one-hot coded outputs corresponding to the ground truth class label (here, $C = 10$).

The first metric is the *information entropy*. For each latent variable, the amount of activation per class c is totaled across the K samples ($K = 10,000$) and normalized

across the C classes to get its class-wise response probabilities

$$P(z_{j,c}) = \frac{1}{\zeta} \sum_{k=1}^{K} z_{jk} y_c \tag{3.20}$$

where ζ is a normalization constant such that the responses sum to 1. The Shannon entropy H_j of each latent variable is then computed as and averaged across the J latent variables ($J = 1,000$) to get

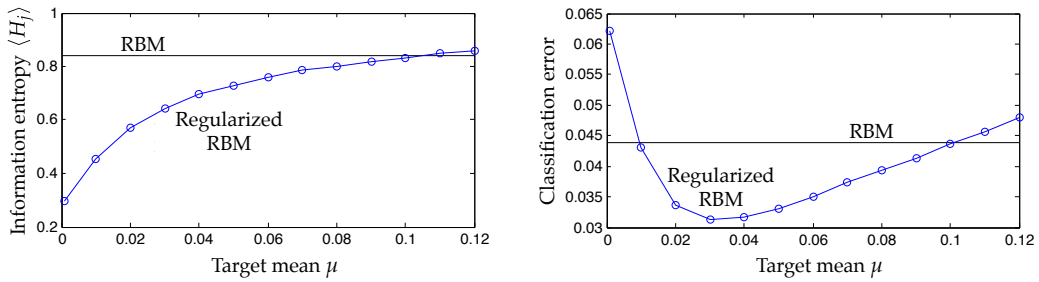
$$\langle H_j \rangle = -\frac{1}{J} \sum_{j=1}^{J} H_j = -\frac{1}{J} \sum_{j=1}^{J} \sum_{c=1}^{C} P(z_{j,c}) \log P(z_{j,c}). \tag{3.21}$$

This metric indicates the level of class-based uncertainty for the latent variables. If $\langle H_j \rangle$ is low, the latent variables encode fewer classes each, while a high $\langle H_j \rangle$ points to each latent variable being shared across more classes.

The second metric is *classification error*, which is the defacto evaluation metric for the MNIST dataset. The class-wise specificity and the image classification results were analyzed for the learned representations with different parameterization of μ. A perceptron with softmax activation units (see Section 2.2.1) was added on top of the latent layer. The perceptron was trained from the activations of the latent layer, without updating the weights through backpropogating, so as to isolate and study the effects of the regularization alone. For evaluation, the classification error rate was computed.

Evaluation results and discussion. From Figure 3.14(a), a monotonic relationship of $\langle H_j \rangle$ with respect to μ was observed. When μ is lowered, the number of instances a latent variable responds to will also decrease, which tends to lower the number of classes the latent variable is selective towards. In the extreme case, when μ is so low such that a latent variable becomes so highly specific that memorizes a single instance [Gross, 2002], which means it also encodes only that one class and $\langle H_j \rangle$ is the lowest.

Figure 3.14(b) shows that the relationship between the classification error and μ is no longer monotonic. For this dataset, the regularized RBM achieves better result than the standard RBM in the approximate range of $0.01 \leq \mu \leq 0.1$. The model has poor generalization when μ nears 0 as units encode individual instances too specifically, but has poor discrimination power when μ is too high. Interestingly, the best classification performance of this simple semi-supervised method without fine-tuning is comparable to other reported results using methods with similar complexity, yet completely supervised approaches [LeCun et al., 1998]. It is, therefore, important to select a suitable regularization method and parameters when performing image classification.



(a) Class-wise information entropy

(b) Classification error

Figure 3.14: Discriminative performance of the regularized RBM. (a) $\langle H_j \rangle$ varies monotonically with μ. (b) Classification error is minimum when μ is low, but not at its lowest. There is a range of μ whereby regularization helps improve the model.

3.4.3 Experiment: modeling color image patches

The objective of this experiment is to model color image patches using an RBM with sparse, selective and topographic regularization. The resulting feature map can be analyzed based on their ability to encode visual appearances within the topographic structure. The model can subsequently be analyzed based on its representations' invariances to various image transformations, such as translation, rotation, scaling, as well as illumination color changes. Thus, a suitable color dataset needs to be selected.

Training dataset and setup. The McGill Calibrated Colour Image Database [Olmos and Kingdom, 2004] was selected for its variety in natural objects (Figure 3.15). A set of 100,000 image patches of size 10×10 were randomly sampled from the images in the dataset. Each patch had three color channels: red, green and blue. The RBM used to model the patches consists of 300 input dimensions (10×10 pixels in 3 color channels) and 400 latent variables. The latent variables were structured in a two-dimensional 20×20 feature map.

Figure 3.15: McGill calibrated color image database. The dataset consists of 9 categories, namely: *"flowers"*, *"animals"*, *"foliage"*, *"textures"*, *"fruits"*, *"landscapes"*, *"winter"*, *"man made"* and *"shadows"*.

The resulting topographic feature map is shown in Figure 3.16(a). Although the learning of grayscale topographic maps have been previously demonstrated by other similar models [Hyvärinen et al., 2001; Welling et al., 2003; Kavukcuoglu et al., 2009], to my knowledge, this is the first topographic feature map that models color visual information. As a comparison, another RBM with 400 latent variables was also trained without topographic regularization. The resulting feature map of this second RBM is shown in Figure 3.16(b).

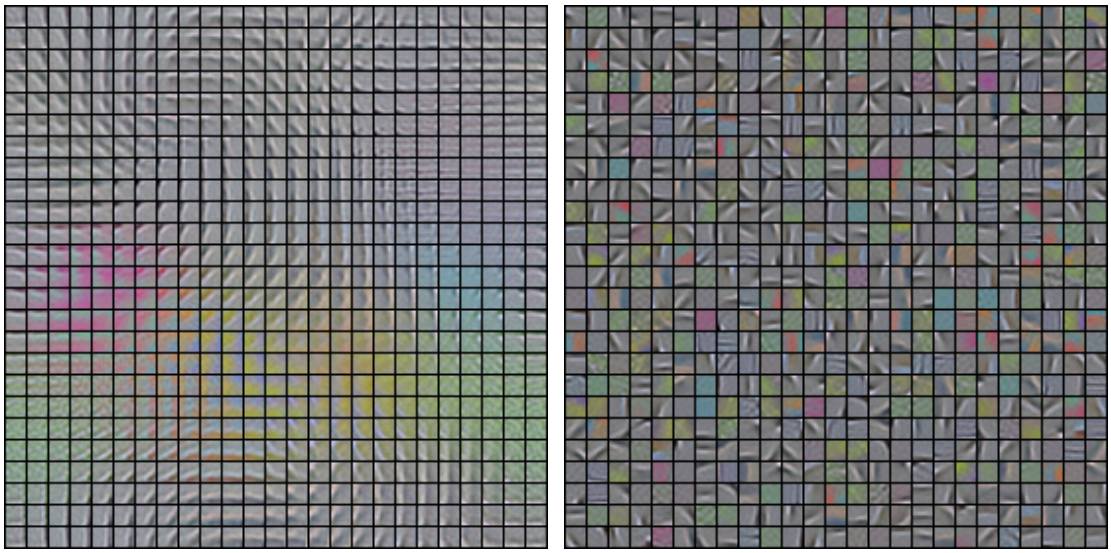
Analysis of topographic feature map. The topographic feature map (Figure 3.16(a)) consists of Gabor-like filters with smoothly varying appearances, as opposed to the feature map without topographic regularization (Figure 3.16(b)). Gabor functions were fitted to the filters of the feature map to separately analyze their visual components of orientation, spatial position and spatial frequency, as presented in Figure 3.17. One observes a smooth transition of appearance between neighboring filters in the feature map for orientation, spatial position and spatial frequency. There is also an emergence of various neurological orientation coding phenomenon, such as orientation pinwheel structures, iso-orientation lines and in the orientation map (Figure 3.17(a)).

The filter map also exhibits spatial clustering by color in terms of the mean saturation of the filter. This is shown in Figure 3.17(b). Some filters exhibit the phenomenon of color-opponency, while others encode a single color. The predominant opponent pairs are red-green, yellow-blue and black-white. Comparing Figure 3.17(b) and Figure 3.17(a), each color-opponent pair has filters with a variety orientation coding. Joint analysis with Figure 3.17(d) indicate that while color-opponent filters tend to be of lower frequency, there are also some single-colored filters – mostly green, cyan and violet – with high frequency textured appearances.

In Figure 3.18, the correlation between neighboring filters is analyzed in terms of different components of their appearance. Strong correlations between neighboring latent variables are observed. For example, Figure 3.18(a) shows the scatter plot of the orientation of a each filter in the x-axis, against the orientations of each of its four neighbors in the y-axis. Many points in the plot fall near to the diagonal axis, which represents a perfect correlation between appearance of a pair of neighbors.[3] A similar correlation is observed for the components of frequency (Figure 3.18(c)) and spatial position (Figure 3.18(b)).[4]

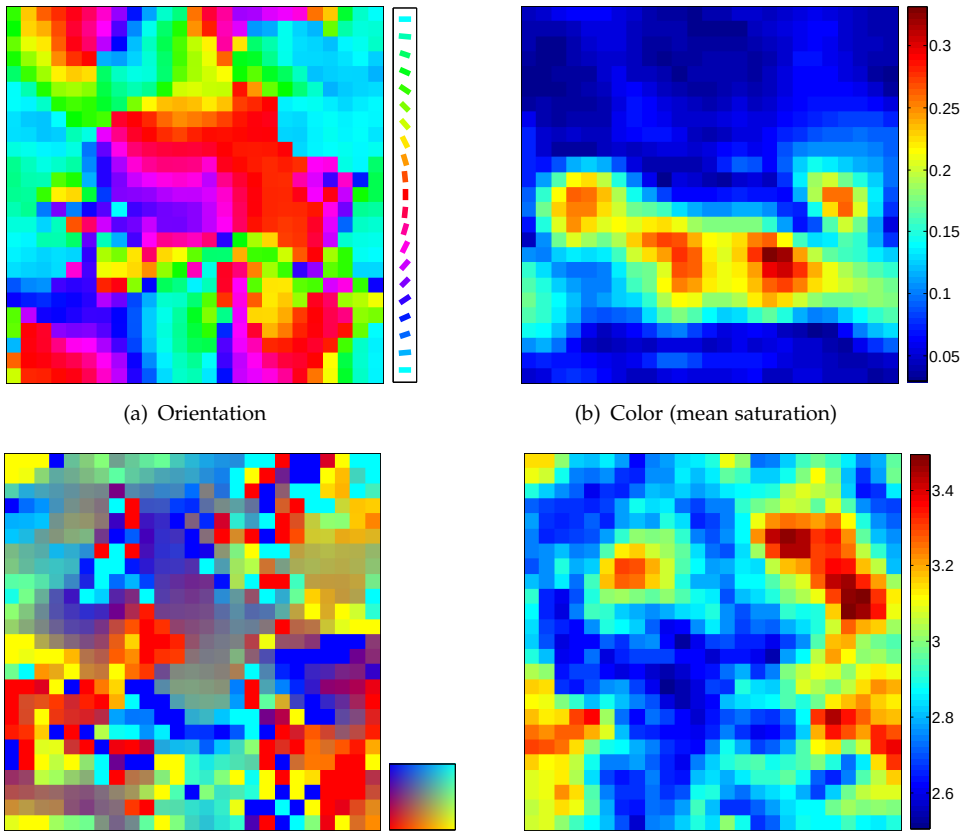
[3]Note that orientation angle from 0 to π wraps around on both axis, thus the points at the top-left and bottom right of the figure indicate a positive correlation.

[4]Note that the spatial position correlation has many boundary cases due to the Gabor fitting method selecting centroid positions way beyond the boundary of the filter size.



(a) Topographic feature map (b) Independent feature map

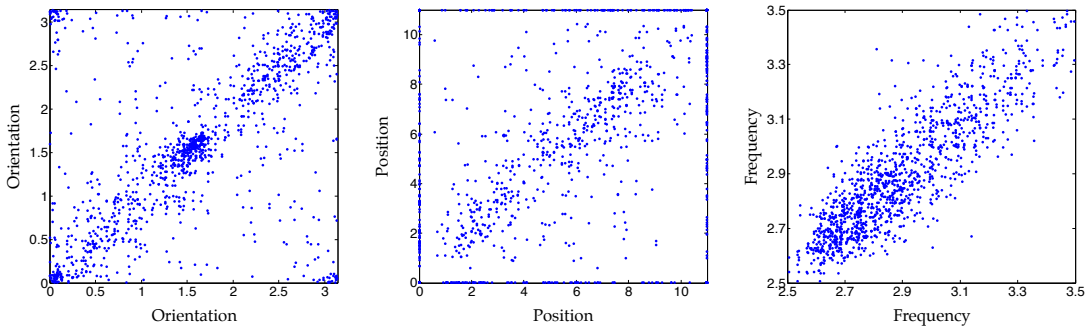
Figure 3.16: Feature maps learned with topographic and independent regularizers. (a) The topographic map has an inherent spatial structure. (b) The independent one appears to be randomly organized.



(a) Orientation (b) Color (mean saturation)

(c) Spatial position (d) Spatial frequency

Figure 3.17: Analysis of appearance components in the topographic feature map. Appearance of filters vary smoothly across the feature map when broken down to their component properties of (a) orientation, (b) color, (c) spatial position and (d) spatial frequency.



(a) Orientation (b) Spatial position (c) Spatial frequency

FIGURE 3.18: Neighboring correlations in the topographic map. The topographic feature map shows strong correlation in appearance between spatial neighbors with respect to each visual component. (a) orientation, (b) spatial position and (c) spatial frequency.

Evaluating representational invariance. The trained model was evaluated based on its invariance to various image transformations (translation, rotation, scaling) and changes to illumination color.

Constructing evaluation datasets. For image transformations, new patches were sampled from the McGill database. Image patches of varying degrees of transformation, relative to a non-transformed patch, were sampled from the datasets. An evaluation batch for rotation consists of 13 patches sampled about a fixed point with rotation ranging from -30 to 30 degrees at 5 degree intervals. The samples for translation were drawn via horizontal translations from -3 to 3 pixels. For scaling, a progressive scaling factor of $1.1\times$ was used to up-sample and down-sample an image patch for 4 scale intervals in either direction.

To evaluate invariance to illumination color, another dataset – the Amsterdam library of object images (ALOI) dataset [Geusebroek et al., 2005] – was used. Specifically, the subset of images used were produced by photographing objects under different illumination color temperature in a controlled environment (see Figure 3.19). To obtain samples with varying illumination color, a set of patches was sampled from the same image coordinate of different images photographed under different illumination temperature (from 2175K to 2975K). For this, 5 warmer and 5 cooler temperature images, relative to the neutral temperature at 2550K, were used.

For each evaluation task, 500 evaluation batches were extracted, each consisting of a set of patches sampled via transformations for the given task. The details of the evaluation batches are summarized in Table 3.4.

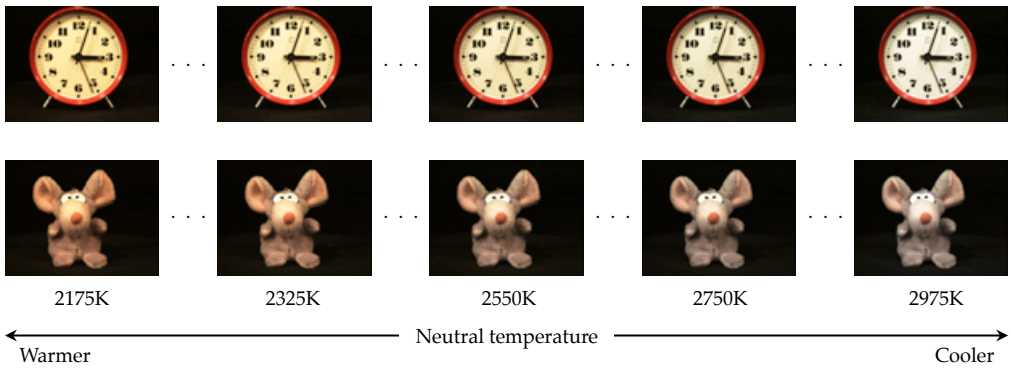


Figure 3.19: Examples of images photographed under different illumination color from the Amsterdam library of object images (ALOI) dataset. Images were photographed under a controlled environment with different illumination temperature ranging from 2175K to 2975K.

Table 3.4: Evaluation batches for invariance analysis.

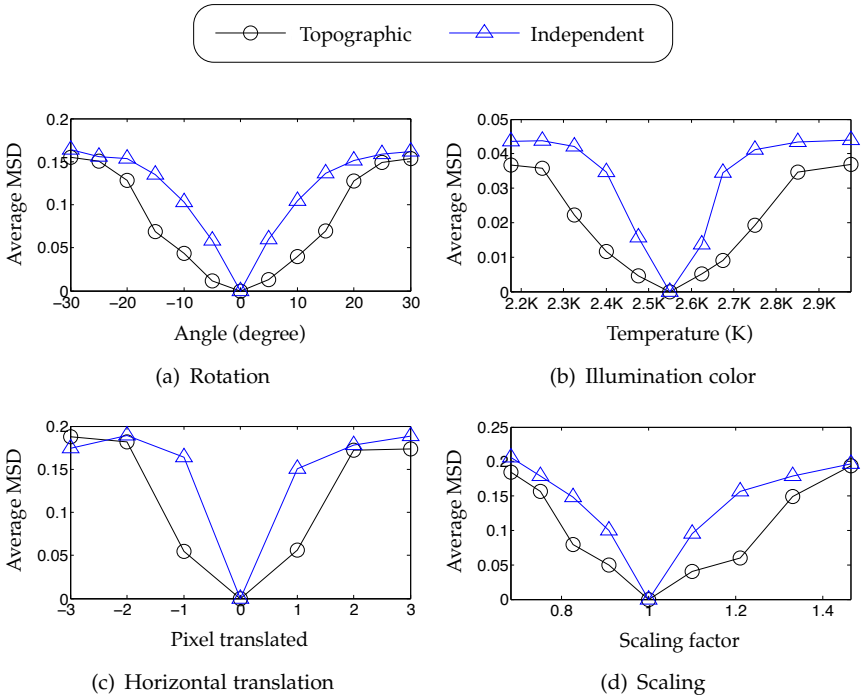
Rotation angle (degree)	Horizontal translation (pixel)	Scaling factor	Illumination temperature (K)	
-30				
-25			$2175K$	
-20		1.1^{-4}	$2250K$	Negative transformation
-15	-3	1.1^{-3}	$2325K$	
-10	-2	1.1^{-2}	$2400K$	
-5	-1	1.1^{-1}	$2475K$	
0	0	1	$2550K$	– No transformation
5	1	1.1	$2625K$	
10	2	1.1^2	$2675K$	
15	3	1.1^3	$2750K$	Positive transformation
20		1.1^4	$2850K$	
25			$2975K$	
30				
13 samples	7 samples	9 samples	11 samples	
McGill database			ALOI dataset	

Evaluation results and discussion. For every input patch, the latent representation was recorded. To quantitively measure invariance of each transformation t in an evaluation batch k, the mean squared difference $\text{MSD}_k(t)$ between the transformed $\mathbf{z}(t)$ and untransformed $\mathbf{z}(0)$ representations. The $\text{MSD}_k(t)$ for the transformation was then averaged across the K evaluation batches ($K = 500$):

$$\text{Average MSD}(t) = \frac{1}{K}\sum_{k=1}^{K}\text{MSD}_k(t) = \frac{1}{JK}\sum_{k=1}^{K}\sum_{j=1}^{J}(z_{jk}(0) - z_{jk}(t))^2. \tag{3.22}$$

The results are plotted in Figure 3.20.

For every type of transformation, when the transformation is low, topographic representations are more invariant than independent ones. This means that the latent representation of a slightly transformed input is a closer match to the untransformed one. As the amount of transformation increases, the representation gradually shifts and the difference between the representations increases. There is little difference between the two models under large transformations, with both producing representations with more differences as the amount of transformation increases.



(a) Rotation (b) Illumination color

(c) Horizontal translation (d) Scaling

FIGURE 3.20: Empirical analysis of representational invariance. Comparing the invariance between representations of the topographic and independent models for (a) rotation, (b) varying illumination color, (c) translation and (d) scaling, the topographic model yields more invariant representation when the transformation is small. When the transformation is large, there is little difference between the models.

3.5 Potential extensions and applications

Due to the generality of the RBM regularization method proposed in Section 3.2, various coding schemes for RBM learning could be introduced. Besides sparsity, selectivity and topographic organization, which have been explored in the previous sections, this section suggests other potential coding schemes and their applications that can be explored.

Explicit coding. Explicitness is semantic property of the coding [Földiák, 2009, 2013]. It describes the relationship between a code, and objects or concepts in the real world. A code is explicit if an individual or a small set of latent variables directly describes

meaningful attributes or categories of the input. In the context of regularization, each coding can be tweaked using supervised labels such that it responds to fewer classes. This can also mean that the class-wise uncertainty based on an information entropy measure can be minimized, which makes this aspect of coding closely related to the second part of the experiments described in Section 3.4.2.

Temporally invariant coding. Another form of invariant coding is temporal invariance. In the real world, given a sequence of images from a video of the same object, the resulting coding should be very similar. This stability in coding across time can be translated to a smoothing regularizer. The coding is related to the trace learning rule, [Földiák, 1991], where different views of the same object can be linked across instances, resulting in invariant object and face recognition [Wallis and Rolls, 97; Rolls and Milward, 2000; Rolls and Stringer, 2001]. The slow feature analysis (SFA) is another of such transformations that help extract temporally-invariant representations [Wiskott and Sejnowski, 2002; Theriault et al., 2013a].

3.6 Summary and discussion

This chapter introduced a new regularization method base on the cross-entropy loss to penalize the activations of each latent variable with respect to each instance. This enables the RBM to learn interesting and desirable representations, while maximizing the likelihood of the data distribution. The desirable representations take the form of a target matrix, which can be designed to encode representational properties, such as sparsity, selectivity and topographic organization, as further proposed in this chapter.

Experiments show that these properties are transferred to the RBM's parameters and are being induced automatically when a new input is being encoded. Upon visualization, the representations learned capture the spatial structure of the image. It is empirically shown that it is important to jointly encode both selectivity and sparsity within the model. This helps improve the discriminative performances, if the parameterization is well selected. Topographic organization can also help to encourage appearance-based invariance.

Being able to manipulate the learning of the representations is an important quality when learning to encode local descriptors in the context of the bag-of-words model. However, it will also be crucial to scale the architectures to handle larger images, rather than merely small images and patches. Chapter 5 describes this integration with the bag-of-words model.

4 Deep Supervised Optimization

Chapter abstract — The starting point for training deep networks is to perform unsupervised learning in a greedy layer-wise manner. Once the structure of the network is formed, the model can be further optimized through supervised learning. Instead of directly transiting to a discriminative error back-propagation algorithm, this chapter proposes the notion of a gradual transition by first integrating bottom-up and top-down information. The learning is performed through a global optimization of all the layers. Two manifestations of this idea are suggested to incorporate top-down information into two different types of deep architectures. The first method is based on the restricted Boltzmann machine, while the second is based on the encoder-decoder network.

The learning algorithm for the deep network based on the restricted Boltzmann machine can be seen as a methodological extension from the previous chapter. This deep learning method, together with the unsupervised regularization method previously proposed, is applied to learn and optimize hierarchical visual dictionaries, described in Chapter 5.

Some of the material in this chapter has been published at the following conference:

- Goh, H., Thome, N., Cord, M., and Lim, J.-H. (2013). Top-down regularization of deep belief networks. *Advances in Neural Information Processing Systems (NIPS)*. [Goh et al., 2013]

4.1 Introduction

Deep architectures have strong representational power due to their hierarchical structures. They are capable of encoding highly varying functions and capturing complex relationships and higher-level abstractions among high-dimensional data. However, the depth of the architecture also makes it challenging to train the entire network through supervised learning due to the sheer number of parameters and the non-convex optimization problem. The supervised optimization of an entire deep network may lead to worse results as compared to shallower networks.

Recent developments in unsupervised feature learning and deep learning algorithms have made it possible to learn deep feature hierarchies.

Deep learning, in its current form, typically involves two separate training phases. The first phase greedily stacks unsupervised modules from the bottom-up and this is followed by a supervised phase that fine-tunes the entire deep network. The unsupervised phase can be seen as a pre-training or initialization step for supervised fine-tuning to take place. The unsupervised learning phase focuses on modeling the input data. For a given set of data, the result of this initialization is constant, subject to parameterization, regardless of the ultimate classification task. This lack of variation means that the supervised phase will need to adapt the parameters of the entire network to suit the classification problem. On the other hand, supervised learning is typically performed using the error backpropagation algorithm, and the parameters of the model are updated based on discriminative signals from the topmost layer.

This chapter proposes a notion to insert an intermediate phase between the two existing phases to have a smoother transition between fully-unsupervised learning and strongly-supervised learning. This is to enable richer cooperation between the bottom-up data and top-down labels.

4.2 Deep supervised fine-tuning: a quick recap

Supervised learning is crucial to optimize the parameters of a deep architecture to be aligned with the classification task (see Section 2.3 for a review). The most popular method of supervised learning is the error backpropagation, which is a fully-supervised method to tune the weights of the network. A well formulated alternative is the up-down algorithm [Hinton et al., 2006], which generatively learns a joint model of the data and its label.

4.2.1 Deep error backpropagation

A common method to introduce supervision to a deep neural network is the error backpropagation algorithm, which is used for training multilayer perceptron (MLP) networks (Section 2.2.1). The algorithm optimizes its parameters based on a discriminative model $P(\mathbf{y}|\mathbf{x})$. After a deep network has been pre-trained through unsupervised learning, the error backpropagation algorithm updates the parameters of the model through gradient descent based on an output error-correcting penalty, which is

typically the cross-entropy loss function for a softmax activated classification layer:

$$\mathcal{L}_{net} = -\sum_{k=1}^{|\mathcal{D}_{train}|}\sum_{c=1}^{C}\log P(y_{ck}|\hat{y}_{ck}), \tag{4.1}$$

where $|\mathcal{D}_{train}| = \{(\mathbf{x}_k \in \mathbb{R}^I, \hat{\mathbf{y}}_k \in \mathbb{R}^C)\}$ is the training dataset and $\hat{\mathbf{y}}_k$ is the output hypothesis given an input. Gradient descent can be performed by backward chaining the partial derivative by projecting the errors in reverse through chain of activations. The problem arises when the error signals become diffused and unusable at the lower layers. Please see Section 2.2.1 for an explanation of this phenomenon. Section 4.4 proposes to address these issues in the framework of a deep encoder-decoder network.

4.2.2 Up-down back-fitting algorithm

Another existing method to fine-tune the parameters of the deep network is the up-down learning algorithm, which is a variant of the "wake-sleep" algorithm [Hinton et al., 1995]. Using a process known as back-fitting, this generative fine-tuning scheme, which models $P(\mathbf{x}, \mathbf{y})$, serves as an alternative to discriminative learning. This method has been used specifically in the deep belief network (DBN) [Hinton et al., 2006]

The pre-training phase of the DBN greedily stacks restricted Boltzmann machines (RBM) to model the input data distribution $P(\mathbf{x})$ through generative unsupervised learning. After being initialized, the undirected weights of the RBMs are untied such that the weights used for recognition and generation are treated as separate entities, except for the final set of weights. The up-down algorithm starts with a bottom-up pass that tunes the generative parameters. The undirected weights of the topmost RBM are then fitted to the posterior distribution of the last layer. Finally, the top-down pass adjusts the recognition connections from the top layer to the bottom layer, such that they are able to perform good recognition.

Supervised restricted Boltzmann machine. Let us zoom in on the top-level RBM to analyze how the labels are incorporated in the DBN. The DBN learns pairs of layers as RBMs, with the top-most layer being a large representational vector. A layer of one-hot coded units is added to the second last layer, such that the inputs of the topmost RBM is a concatenation of the latent representation $\mathbf{z}_{L-1}$ and the output vector $\mathbf{y}$, as shown in Figure 4.1. This method of incorporating labels is also used in the generative portion of a hybrid RBM that combines generative and discriminative optimizations [Larochelle and Bengio, 2008].

The latent layer $\mathbf{z}_L$ of the final RBM $\mathbf{W}_{L-1}$, models the interactions between units of its input layer, which is made up of the previous layer of latent variables $\mathbf{z}_{l-1}$ and outputs

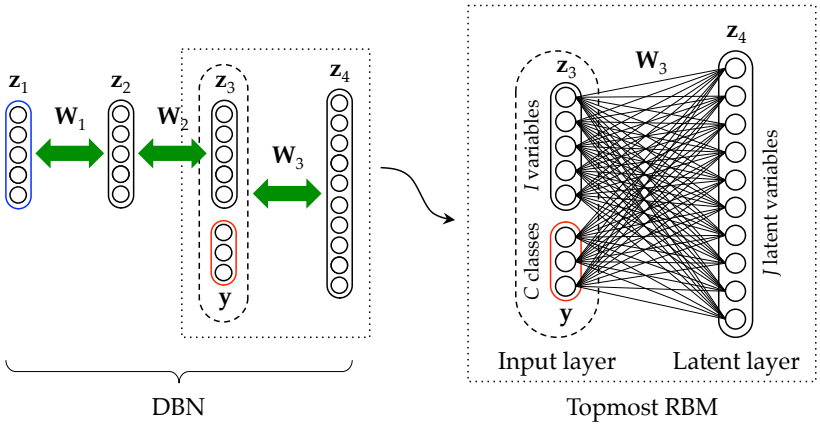


Figure 4.1: Incorporating class labels into a deep belief network (DBN). A DBN is made up of a stack of restricted Boltzmann machines (RBM). Class labels are included as additional inputs to the topmost RBM.

$\mathbf{y}$. The greedy learning of a stack of RBMs $\{\mathbf{W}_1, \ldots, \mathbf{W}_{L-2}\}$, improves the variational bound on likelihood of $P(\mathbf{x})$ [Neal and Hinton, 1998; Hinton et al., 2006], which is eventually well modeled by $\mathbf{z}_{l-1}$. By learning a joint representation of $\mathbf{z}_{L-1}$ and $\mathbf{y}$, this final RBM essentially models the joint distribution $P(\mathbf{x}, \mathbf{y})$.

During training, when both $\mathbf{x}$ and $\mathbf{y}$ are known, the probability of activating a unit in the topmost layer is given by

$$P(z_{L,j} = 1 \mid \mathbf{z}_{L-1}, \mathbf{y}; \mathbf{W}_{L-1}) = \frac{1}{1 + \exp(-\sum_{i=0}^{I} w_{L-1,ij} z_{L-1,i} - \sum_{c=1}^{C} w_{L-1,cj} y_c)}. \quad (4.2)$$

However, during inference, $\mathbf{y}$ is not given and the units are set to a "noisy" neutral state. The activation probability of $\mathbf{y}$ can be computed through backward sampling via softmax units:

$$P(y_c = 1 \mid \mathbf{z}_L; \mathbf{W}_{L-1}) = \frac{\exp(\sum_{j=0}^{J} w_{L-1,cj} z_{L,j})}{\sum_{q=1}^{C} \exp(\sum_{j=0}^{J} w_{L-1,qj} z_{L,j})}. \quad (4.3)$$

Multiple iterations of forward-backward activations could then be performed to generate more accurate samples of $\mathbf{y}$. The inference process basically attempts to "denoise" and stabilizes the joint representation, given a "noisy" output vector $\mathbf{y}$. Performance may suffer if the number of classes is huge, which results in a low signal-to-"noise" ratio, which may be particularly difficult to "denoise".

The proposal of this chapter is to extend this idea of a joint bottom-up and top-down learning to get a gradual transition between the learning phases. Two implementations of this principle are presented: one is based on the RBM and the other is based on the encoder-decoder network. Section 4.3 suggests a RBM-regularization-based method to incorporate class labels during the fine-tuning phase. Meanwhile, Section 4.4 realizes the hybridization using reconstructive and predictive loss functions.

4.3 Top-down regularized deep belief network

Starting from the initialization of the DBN as a stack of greedily learned RBMs, this section proposes a new method to incorporate top-down supervision into the optimization. The method is an extension of the regularization method previously proposed in Section 3.3.1 that extends the contrastive divergence learning algorithm.

4.3.1 Top-down regularization: the basic building block

Previously in Section 3.3.1, I introduced the method for point- and instance-wise regularization of RBMs using target latent activations based on inductive biases that might suit the task, such as sparsity, selectivity and topographic organization. Now, the same principle of regularizing the latent activations can be used to combine signals from the bottom-up and top-down. This forms the building block for optimizing a DBN with top-down regularization.

The basic building block is a three-layer structure consisting of three consecutive layers: the previous $\mathbf{z}_{l-1} \in \mathbb{R}^I$, current $\mathbf{z}_l \in \mathbb{R}^J$ and next $\mathbf{z}_{l+1} \in \mathbb{R}^H$ layers. The layers are connected by two sets of weight parameters $\mathbf{W}_{l-1}$ and $\mathbf{W}_l$ to the previous and next layers respectively. For the current layer $\mathbf{z}_l$, the bottom-up representations $\mathbf{z}_{l,l-1}$ are sampled from the previous layer $\mathbf{z}_{l-1}$ through weighted connections $\mathbf{W}_{l-1}$.

$$P(z_{l,l-1,j} = 1 \mid \mathbf{z}_{l-1}; \mathbf{W}_{l-1}) = \frac{1}{1 + \exp(-\sum_{i=0}^{I} w_{l-1,ij} z_{l-1,i})}, \tag{4.4}$$

where the two terms in the subscripts of a sampled representation $\mathbf{z}_{dest,src}$ refer to the destination (*dest*) and source (*src*) layers respectively. Meanwhile, sampling from the next layer $\mathbf{z}_{l+1}$ via weights $\mathbf{W}_l$ drives the top-down representations $\mathbf{z}_{l,l+1}$:

$$P(z_{l,l+1,j} = 1 \mid \mathbf{z}_{l+1}; \mathbf{W}_l) = \frac{1}{1 + \exp(-\sum_{h=0}^{H} w_{l,jh} z_{l+1,h})}. \tag{4.5}$$

The objective is to learn the RBM parameters $\mathbf{W}_{l-1}$ that map from the previous layer $\mathbf{z}_{l-1}$ to the current latent layer $\mathbf{z}_{l,l-1}$, by maximizing the likelihood of the previous layer $P(\mathbf{z}_{l-1})$ while considering the top-down samples $\mathbf{z}_{l,l+1}$ from the next layer $\mathbf{z}_{l+1}$ as target representations. Following the method proposed in Section 3.3.1, this can be realized as a cross-entropy-regularized maximum likelihood optimization:

$$\mathcal{L}_{l,RBM+top-down} = -\sum_{k=1}^{|\mathcal{D}_{train}|} \log P(\mathbf{z}_{l-1,k}) - \lambda \sum_{k=1}^{|\mathcal{D}_{train}|} \sum_{j=1}^{J} \log P(z_{l,l+1,jk} | z_{l,l-1,jk}). \tag{4.6}$$

This results in the following gradient descent update:

$$\Delta w_{l-1,ij} = \varepsilon \left(\langle z_{l-1,l-2,i} s_{l,j} \rangle_{data} - \langle z_{l-1,l,i} z_{l,l-1,j} \rangle_{recon} \right), \tag{4.7}$$

where

$$s_{l,jk} = \underbrace{(1-\phi_l) z_{l,l-1,jk}}_{\text{Bottom-up}} + \underbrace{\phi_l z_{l,l+1,jk}}_{\text{Top-down}}, \tag{4.8}$$

is the merged representation from the bottom-up and top-down signals, weighted by hyperparameter ϕ_l. The bias towards one source of signal can be adjusted by selecting an appropriate ϕ_l. Additionally, the alternating Gibbs sampling, necessary for the contrastive divergence updates, is performed from the unbiased bottom-up samples using Equation 4.4 and a symmetric decoder:

$$P(z_{l-1,l,j} = 1 \mid \mathbf{z}_{l,l-1}; \mathbf{W}_{l-1}) = \frac{1}{1 + \exp(-\sum_{j=0}^{J} w_{l-1,ij} z_{l,l-1,j})}. \tag{4.9}$$

Figure 4.2 describes the framework for the top-down driven regularization of the RBM – the basic building block for a top-down regularized DBN.

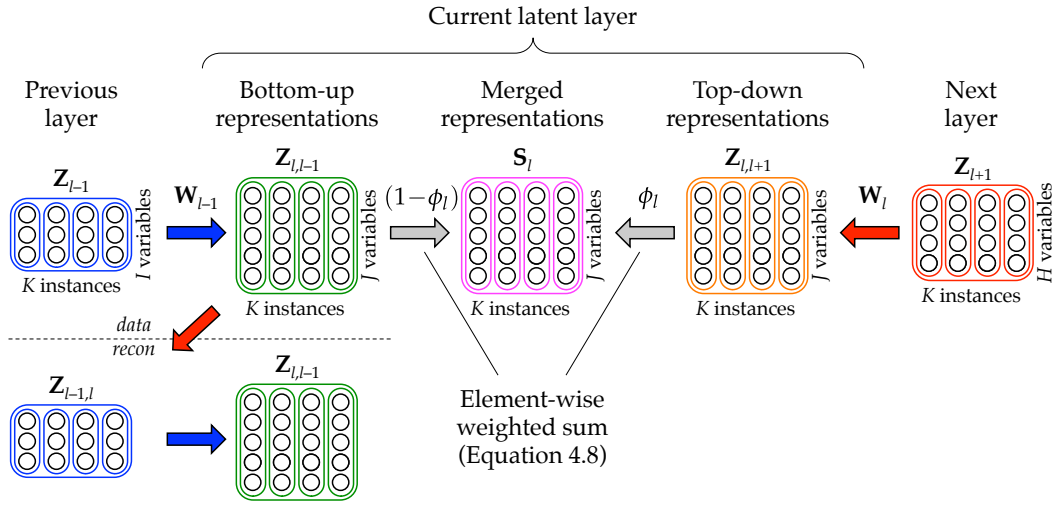


Figure 4.2: Top-down restricted Boltzmann machine (RBM) regularization using a training batch. Bottom-up $\mathbf{Z}_{l,l-1}$ and top-down $\mathbf{Z}_{l,l+1}$ latent representations are sampled from the previous $\mathbf{Z}_{l-1}$ and next $\mathbf{Y}$ layers respectively. They are linearly weighted using the modulatory weight ϕ_l to get the merged representations $\mathbf{S}_l$ of the current latent layer, used for parameter updates. Reconstructions sampled independently from the bottom-up signals form the Gibbs sampling Markov chain.

4.3.2 Constructing a top-down regularized deep belief network

Forward-backward learning strategy. A deep network is constructed by stacking the basic building blocks (Figure 4.3(a)) in a layer-wise hierarchy. The network, as illustrated in Figure 4.3(b), comprises of a total of $L-1$ RBMs. The network can be

trained with a forward and backward strategy (Figure 4.3(c)), similar to the DBN. It integrates top-down regularization with contrastive divergence learning, which is given by alternating Gibbs sampling between the layers (Figure 4.3(d)).

Forward pass and backward pass. Let's consider a DBN with L layers. In the forward pass, given the input features, each layer $\mathbf{z}_l$ is sampled from the bottom-up, based on the representation of the previous layer $\mathbf{z}_{l-1}$ (Equation 4.4). Upon reaching the topmost layer, the backward pass begins. The top layer activations $\mathbf{z}_L$ are combined with the output labels $\mathbf{y}$ to produce $\mathbf{s}_L$ given by

$$s_{L,ck} = \underbrace{(1-\phi_L) z_{L,L-1,ck}}_{\text{Bottom-up}} + \underbrace{\phi_L y_{ck}}_{\text{Class label}}, \tag{4.10}$$

The merged activations $\mathbf{s}_l$ (Equation 4.8), which besides being used for parameter updates, have a second role of activating the lower layer $\mathbf{z}_{l-1}$ from the top-down:

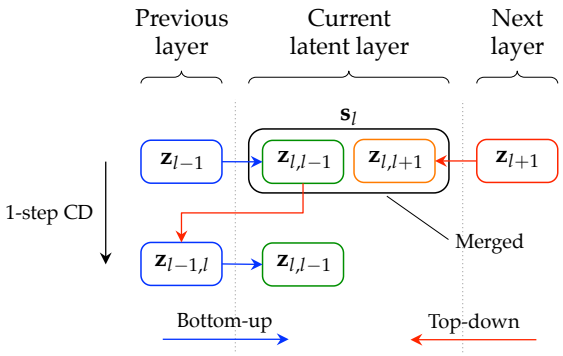
$$P(z_{l-1,l,j} = 1 \mid \mathbf{s}_l; \mathbf{W}_l) = \frac{1}{1 + \exp(-\sum_{h=0}^{H} w_{l-1,jh} s_{l,h})}. \tag{4.11}$$

This is repeated until the second layer is reached ($l = 2$) and $\mathbf{s}_2$ is computed.

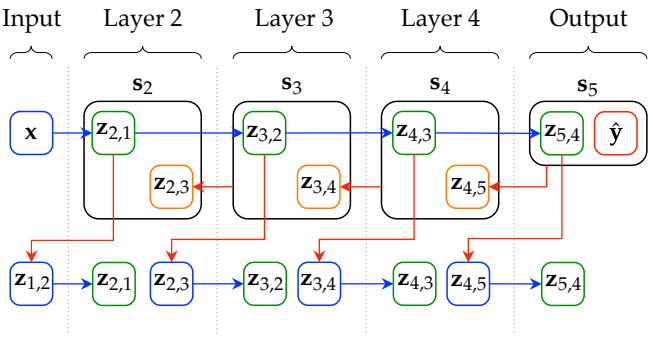
Top-down sampling drives the class-based invariance and clustering of the bottom-up representations. However, if sampling is done directly from the output vector $\mathbf{y}$ alone to the bottom-layers, there will only be one activation pattern per class for each layer. This may be too invariant, especially for the bottom layers. By merging the top-down representations with the bottom-up ones, the representations also encode instance-based variations that cannot be produced through direct sampling from $\mathbf{y}$. Only in the last layer, ϕ_L is usually set as 1 (see next page), so that the final RBM $\mathbf{W}_{L-1}$ learns to map to the class labels $\mathbf{y}$. Backward activation of $\mathbf{z}_{L-1,L}$ is a class-based invariant representation obtained from $\mathbf{y}$ and used to regularize $\mathbf{W}_{L-2}$. All other backward activations from this point on are based on the merged representation from instance- and class-based representations (Figure 4.4).

Alternating Gibbs sampling In addition, using the data-sampled activations computed in the forward pass, alternating Gibbs sampling can be performed concurrently using Equation 4.9 and Equation 4.4 to generate the backward and forward reconstructions respectively. This is necessary for the contrastive divergence learning and is similar to the up-pass of the DBN up-down learning algorithm.

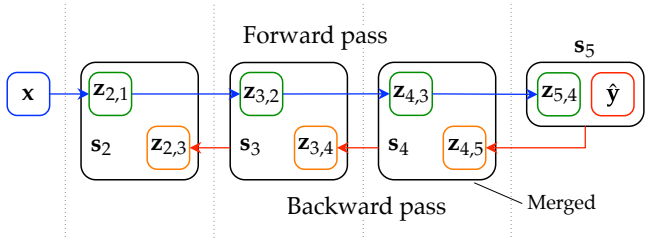
Parameter updates. After the forward and backward passes and Gibbs sampling, all $L-1$ RBMs in the network can be updated at the same time using Equation 4.7. Algorithm 4.1 describes this training strategy when using mini-batches to optimize the entire deep network.



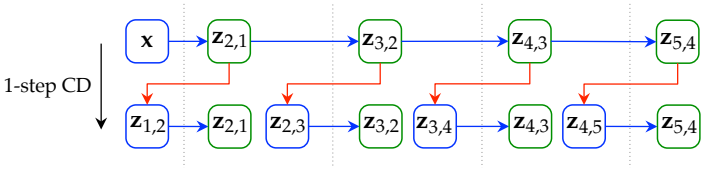
(a) Basic building block.



(b) Top-down regularized deep belief network.



(c) Forward and backward passes for top-down regularization.



(d) Alternating Gibbs sampling chains for contrastive divergence learning.

Figure 4.3: Constructing a top-down regularized deep belief network (DBN). All the restricted Boltzmann machines (RBM) that make up the network are concurrently optimized. (a) A building block spans three layers and consists of an RBM trained with contrastive divergence regularized by top-down signals. (b) The building blocks are connected layer-wise. Both bottom-up and top-down activations are used for training the network. (c) Activations for the top-down regularization are obtained by sampling and merging the forward pass and the backward pass. (d) From the activations of the forward pass, the reconstructions can be obtained by performing alternating Gibbs sampling with the previous layer.

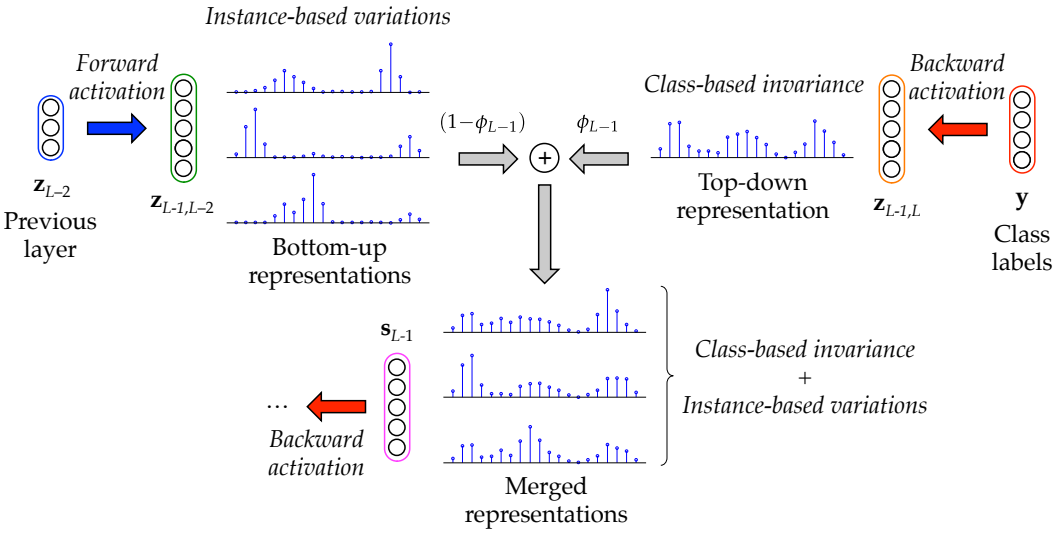


FIGURE 4.4: Merging class-based invariance and instance-based variations. In the second last layer, class-based representations are obtained by sampling from the output layer. Instance-based variations are created by merging the representations, which are then used for subsequent backward pass activations.

Algorithm 4.1: Top-down regularized deep belief network

for $l = 1$ *to* $L-1$ **do** // Greedy RBM learning (Section 3.3)
  Initialize $\mathbf{W}_l$
end
Get $\mathbf{Z}_{1,0,0} \leftarrow \mathbf{X}_0$ from randomized training batch
repeat
  for $l = 2$ *to* L **do** // Forward pass
    Sample bottom-up activations: $P_0(\mathbf{Z}_{l,l-1,0}|\mathbf{Z}_{l-1,l-2,0})$ // Equation 4.4
  end
  Merge top-layer representations: $\mathbf{S}_L$ // Equation 4.10
  for $l = L-1$ *to* 2 **do** // Backward pass
    Sample top-down activations: $P(\mathbf{Z}_{l,l+1}|\mathbf{S}_{l+1})$ // Equation 4.11
    Compute merged representations: $\mathbf{S}_l$ // Equation 4.8
  end
  for $l = 1$ *to* $L-1$ **do**
    for $n = 1$ *to* N **do** // Gibbs sampling
      Sample $P_n(\mathbf{Z}_{l,l+1,n}|\mathbf{Z}_{l+1,l,n-1})$ // Equation 4.4
      Sample $P_n(\mathbf{Z}_{l+1,l,n}|\mathbf{Z}_{l,l-1,n})$ // Equation 4.5
    end
    Update $w_{l,ij} := w_{l,ij} + \Delta w_{l,ij}$ // Equation 4.7
  end
until *convergence*

Setting the modulation parameter. To learn a gradual mapping between the inputs and outputs, their influences on the intermediate layers should be gradually modulated. Layers nearer to the input should have a higher bias towards the input, while layers closer to the output should be biased more by the output. One way to achieve this is to set ϕ_l based on the distance of the latent layer $\mathbf{z}_l$ to the output

layer. The modulatory weight ϕ_l, which stipulates how much influence the top-down information has on the learning of weight $\mathbf{W}_{l-1}$, may be assigned as follows:

$$\phi_l = \frac{\left|l-1\right|^{\zeta}}{\left|l-1\right|^{\zeta} + \left|L-l\right|^{\zeta}}, \tag{4.12}$$

where $\zeta > 0$ is a single hyperparameter that controls the behavior of ϕ_l across the layers and replaces the need to tune them individually. If $\zeta = 1$, then ϕ_l is linearly assigned across the layers. When ζ increases, then the bias towards the nearer source grows logarithmically. Table 4.2 shows the values of ϕ_l for the five-layer deep network, similar to the one shown in Figure 4.3(b), for various settings of ζ. Since ϕ_L always assumes the value of 1, then, from Equation 4.10, $\mathbf{s}_L = \mathbf{y}$ and $\mathbf{W}_{L-1}$ is trained as an associative memory to the outputs $\mathbf{y}$ against its equilibrium distribution:

$$\Delta w_{L-1,ic} = \varepsilon \left(\langle z_{L-1,i} y_{l,c} \rangle_{data} - \langle z_{L-1,i} z_{L,L-1,c} \rangle_{recon} \right). \tag{4.13}$$

Table 4.2: Modulatory weights ϕ_l in relation to hyperparameter ζ (for $L = 5$).

ζ	ϕ_2	ϕ_3	ϕ_4	ϕ_5
0.1	0.47	0.50	0.53	1.00
0.5	0.37	0.50	0.63	1.00
1.0	0.25	0.50	0.75	1.00
2.0	0.10	0.50	0.90	1.00
3.0	0.04	0.50	0.96	1.00
4.0	0.01	0.50	0.99	1.00

4.3.3 Three-phase deep learning strategy

While a greedy layer-wise stacking of RBMs generatively models $P(\mathbf{x})$, the eventual goal of the network is to give a prediction of $P(\mathbf{y}|\mathbf{x})$. The discriminative optimization of $P(\mathbf{y}|\mathbf{x})$ is given by the error backpropagation algorithm. As a result, it is still important to perform learning in a generative and discriminative manner. The proposed top-down regularization algorithm (Section 4.3.2) is perfectly suited to obtain a smoother transition between the two training phases. This suggests the adoption of a three-phase strategy for training a deep network, whereby the parameters learned in one phase initializes the next, as follows:

- *Phase 1 - unsupervised.* Greedy layer-wise stacking.
- *Phase 2 - supervised regularized.* Forward-backward deep learning.
- *Phase 3 - supervised discriminative.* Classification error backpropagation.

Phase 1: unsupervised. In Phase 1, the network is constructed by greedily learning new RBMs on top of the existing network. To enhance the representations learned through this unsupervised process, various regularization methods can be applied, such as the jointly sparse and selective regularization that was introduced in Chapter 3.3.2. The stacking process is repeated for $L-2$ RBMs, until layer $L-1$ is added to the network. Please see Section 2.3.1 for a detailed discussion.

Phase 2: supervised regularized. Phase 2 begins by connecting the $L-1$ to a final layer, with the number of units corresponding to the number of class labels. For a classification problem, the final layer can be activated by the softmax activation function. The RBM projecting to this layer is first learned by using the one-hot coded output vector $\mathbf{y} \in \mathbb{R}^C$ as its target activations in its regularization and by setting modulatory weights ϕ_L to 1, which results in the parameter update given by Equation 4.13. This final RBM, together with the parameters of other RBMs in the network learned from Phase 1, form the initialization for performing deep learning with top-down regularization using the forward-backward algorithm, as explained in the earlier section. This phase fine-tunes the network and binds the layers together by aligning all the parameters of the network with the output labels.

Phase 3: supervised discriminative. Finally, Phase 3 takes the existing parameters of the network and applies the error backpropagation algorithm to explicitly improve class discrimination in the representations. Similar to the previous phase, backpropagation can be separated into a forward pass followed by a backward pass. In the forward pass, each layer is activated from the bottom-up to obtain class predictions of the input. The classification error is then computed based on the ground truth. Finally, the backward pass performs gradient descent on the parameters of the deep network by backpropagating the errors through the layers. Although the same limitations of a diffused gradient in the lower layers still exist, they have been alleviated by the learning performed in the first two phases that presets the model to regions with good local minima.

From Phase 1 to Phase 2, the form of the parameter update rule based on gradient descent does not change. Only that top-down signals are also taken into account. Essentially, the two phases are performing a variant of the contrastive divergence algorithm. Meanwhile, from Phase 2 to Phase 3, the top-level output vector remains unchanged.

4.4 Predictive and reconstructive encoder-decoders

As earlier mentioned, this chapter proposes the notion of performing a hybrid bottom-up and top-down learning to bridge the gap between fully-unsupervised learning and strongly-supervised learning. Two implementations are suggested for this fine-tuning phase, based on 1) the restricted Boltzmann machine and 2) the encoder-decoder networks as the basic building blocks. In the previous section, I have introduced a method to globally align a deep belief network to model the labels. In this section, I will explore the notion of a globally-optimized deep learning algorithm using a deep network based on the encoder-decoder network. Please refer to Sections 2.2.2 and 2.3.2 for a detailed introduction of the encoder-decoder network and its extension to a deep network.

4.4.1 Bottom-up and top-down loss functions

As previously introduced (Section 2.3.2), the deep encoder-decoder network is made up of a stack of encoder-decoder networks, each with the latent coding optimized to reconstruct the representations of the previous layer, while also learning the parameters to project to this latent layer. When the last layer is reached, the error backpropagation algorithm is typically performed to optimize the entire network. This section explores possible alternatives for incorporating supervised labels into the encoder-decoder network.

Given a stack of encoder-decoder networks that is bound together by the common coding of each layer, as shown in Figure 4.5, the loss function for the entire network given a training example, can be expressed as a combination of layer-wise loss functions as follows:

$$\mathcal{L}_{net} = \sum_{l=1}^{L} \mathcal{L}_l \tag{4.14}$$

where $\mathcal{L}_l$ is the loss function for the representation of layer l.

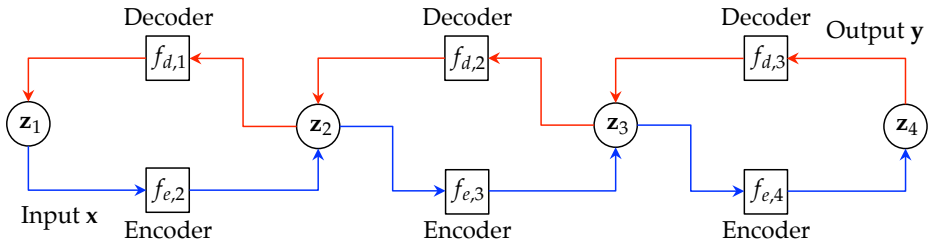


Figure 4.5: A stack of encoder-decoder networks bound by the layers. For a network with L layers, a stack of $L-1$ encoder-decoder networks is required. Each intermediate latent layer is shared by two successive encoder-decoder networks.

When learning a mapping between input layer $\mathbf{x}$ and output layer $\mathbf{y}$, every latent layer l should have the following five properties:

1. *Sparse and selective.* Layer l should be sparse and selective (Section 3.2.1).
2. *Predictive.* The output layer $\mathbf{y}$ can be predicted from layer l.
3. *Reconstructive.* The input layer $\mathbf{x}$ can be reconstructed from layer l.
4. *Encoding.* Layer l can be encoded, given the first layer $\mathbf{z}_1$.
5. *Decoding.* Layer l can be decoded from the last layer $\mathbf{z}_L$.

From this, the SPREAD optimization[1] is proposed. Each layer in the network has a loss function that considers all five requirements:

$$\mathcal{L}_l = \underbrace{\lambda_{s,l} h(\mathbf{z}_l)}_{\text{sparse \& selective}} + \underbrace{\lambda_{p,l} \|\mathbf{y} - \hat{\mathbf{z}}_{L,l}\|^2}_{\text{predictive}} + \underbrace{\lambda_{r,l} \|\mathbf{x} - \hat{\mathbf{z}}_{1,l}\|^2}_{\text{reconstructive}} + \underbrace{\lambda_{e,l} \|\mathbf{z}_l - \hat{\mathbf{z}}_{l,1}\|^2}_{\text{encoding}} + \underbrace{\lambda_{d,l} \|\mathbf{z}_l - \hat{\mathbf{z}}_{l,L}\|^2}_{\text{decoding}} \tag{4.15}$$

where $h(\cdot)$ is a sparse and selective regularizer (see Section 3.3). Here, $\hat{\mathbf{z}}_{m,n}$ represents the coding of a destination layer m that is propagated from the coding $\mathbf{z}_n$ of a source layer n defined as:

$$\hat{\mathbf{z}}_{m,n} = \begin{cases} \mathbf{z}_n & \text{if } n = m \\ f_{e,m}(\dots f_{e,n+1}(\mathbf{z}_n, \mathbf{W}_n)\dots), \mathbf{W}_{m-1}) & \text{if } m > n \text{ (forward)} \\ f_{d,m}(\dots f_{d,n-1}(\mathbf{z}_n, \mathbf{V}_n)\dots), \mathbf{V}_{m+1}) & \text{if } m < n \text{ (backward)} \end{cases} \tag{4.16}$$

where $f_{e,m}(\cdot, \mathbf{W}_{m-1})$ and $f_{d,m}(\cdot, \mathbf{V}_{m+1})$ represent the forward encoding and backward decoding activation functions respectively to obtain the representation of layer m through weights $\mathbf{W}_{m-1}$ and $\mathbf{V}_{m+1}$. The penalties for the prediction, reconstruction, encoding and decoding operations can be visualized with the factor graph of Figure 4.6. Alternatively, the cross entropy loss can be used for the predictive errors, depending on the activation function of the final layer.

For a fully connected network, the activation functions $f(\cdot)$ are usually a squashing function such as the sigmoid or tanh functions, although it is also common not to introduce any nonlinearity in the activations, as in Ranzato et al. [2006]. The hyperbolic tangent $\tanh(\cdot)$ function is used for this discussion:

$$\hat{\mathbf{z}}_{l,l-1} = f_{e,l}(\mathbf{z}_{l-1}, \mathbf{W}_{l-1}) = \tanh(\mathbf{W}_{l-1}^T \mathbf{z}_{l-1}), \tag{4.17}$$

$$\hat{\mathbf{z}}_{l,l+1} = f_{d,l}(\mathbf{z}_{l+1}, \mathbf{V}_{l+1}) = \tanh(\mathbf{V}_{l+1} \mathbf{z}_{l+1}). \tag{4.18}$$

[1] SPREAD is the abbreviation for **S**parse/selective, **P**redictive, **R**econstructive, **E**ncoding, **A**nd **D**ecoding, as defined by the layer-wise loss function.

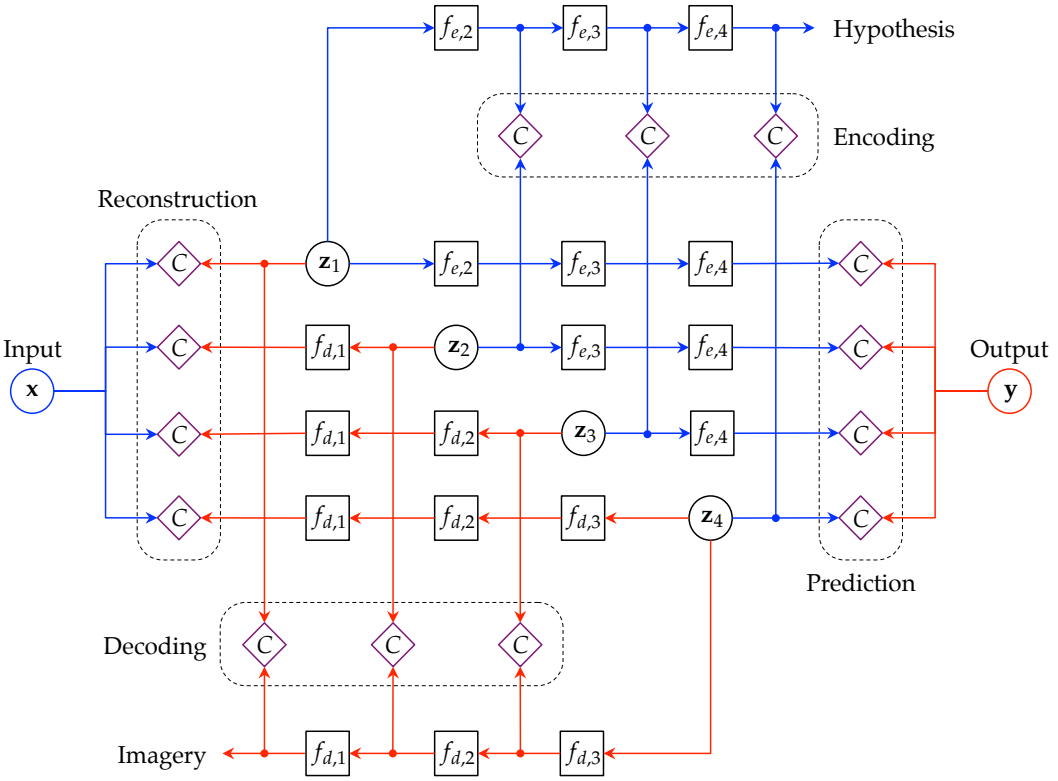


Figure 4.6: Factor graph for the SPREAD optimization. The factor graph shows four distinct sets of costs (C), for prediction, reconstruction, encoding and decoding errors. Each layer is connected to each of the four costs either directly or via a forward or backward chain of activations. For simplicity, the costs for sparsity and selectivity have been excluded from this illustration.

Other differentiable operators may also be included in the chain of feedforward encoders and feedback decoders, depending on the requirements of the task. For vision related tasks, the operators may include convolution (see Section 2.3.3). Another common variation is to share the weights between the feedforward and feedback activations, such that the connections between layers become symmetric $\mathbf{W}_l = \mathbf{V}_{l+1}$.

4.4.2 Globally optimized deep learning

The SPREAD optimization can be split into two steps that alternates between learning the codes $\{\mathbf{z}_l : 1 \leq l \leq L\}$ and training the parameters $\{\mathbf{W}_l : 1 \leq l < L\}$ and $\{\mathbf{V}_l : 1 < l \leq L\}$. A divide and conquer strategy can be adopted by using two steps to optimize loss function in parts. The first step is a *code optimization* step. It finds optimal codes $\mathbf{z}_l^*$ by concentrating on enabling each layer to be able to predict the outputs $\mathbf{y}$ and reconstruct the inputs $\mathbf{x}$. The second step performs *dictionary learning*, which attempts to obtain these optimal codings from the input and output layers. In both steps, error backpropagation is employed.

Step 1: code optimization. The focus of this step is to find optimal codes $\mathbf{z}_l^*$ that are able to perform prediction of the output layer and reconstruction of the input layer. Hence, the weights of $\lambda_{e,l}$ and $\lambda_{d,l}$ are set to zero. This is similar to the decoding step in decoder networks. Essentially, each layer can be optimized independently with a loss function consisting only of the terms for sparsity and selectivity, prediction and reconstruction (Figure 4.7(a)):

$$\mathcal{L}_l^{(1)} = \underbrace{\lambda_{s,l} h(\mathbf{z}_l)}_{\text{sparse \& selective}} + \underbrace{\lambda_{p,l} \|\mathbf{y} - \hat{\mathbf{z}}_{L,l}\|^2}_{\text{predictive}} + \underbrace{\lambda_{r,l} \|\mathbf{x} - \hat{\mathbf{z}}_{1,l}\|^2}_{\text{reconstructive}} \tag{4.19}$$

For fixed parameters $\mathbf{W}_l$ and $\mathbf{V}_l$, the optimal codes $\mathbf{z}_l^*$ that optimize every $\mathbf{z}_l$ can be found using gradient descent:

$$\frac{\partial \mathcal{L}_{net}^{(1)}}{\partial \mathbf{z}_l} \propto \lambda_{s,l} h'(\mathbf{z}_l) - \lambda_{p,l}(\mathbf{y} - \hat{\mathbf{z}}_{L,l}) \frac{\partial \hat{\mathbf{z}}_{L,l}}{\partial \mathbf{z}_l} - \lambda_{r,l}(\mathbf{x} - \hat{\mathbf{z}}_{1,l}) \frac{\partial \hat{\mathbf{z}}_{1,l}}{\partial \mathbf{z}_l} \tag{4.20}$$

Additionally, since the final task is to produce a hypothesized label of $\arg\max_c \hat{\mathbf{y}}$ given an input $\mathbf{x}$, then $\lambda_{p,1}$ and $\lambda_{s,1}$ may be set to zero. This will result in the optimal $\mathbf{z}_1$ to be $\mathbf{x}$, unless a model-based distortion of the input is deemed to be beneficial. Similarly, setting $\lambda_{r,L}$ to zero is optional, depending on whether an instance-dependent relaxation on the output variables is desirable.

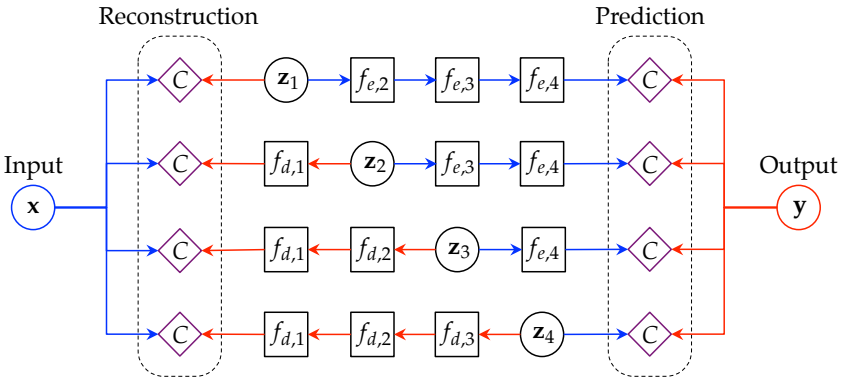
Step 2: deep dictionary learning. The objective of step 2 is to learn the parameters that map to the desired representations that have been found in the first step. This also removes the need for code optimization after learning, when performing the tasks. The regularization parameters of $\lambda_{s,l}$, $\lambda_{p,l}$ and $\lambda_{r,l}$ are now set to zero, resulting in the following loss (Figure 4.7(b)):

$$\mathcal{L}_{net}^{(2)} = \sum_{l=1}^{L} \underbrace{\lambda_{e,l} \|\mathbf{z}_l - \hat{\mathbf{z}}_{l,1}\|^2}_{\text{encoding}} + \underbrace{\lambda_{d,l} \|\mathbf{z}_l - \hat{\mathbf{z}}_{l,L}\|^2}_{\text{decoding}} \tag{4.21}$$

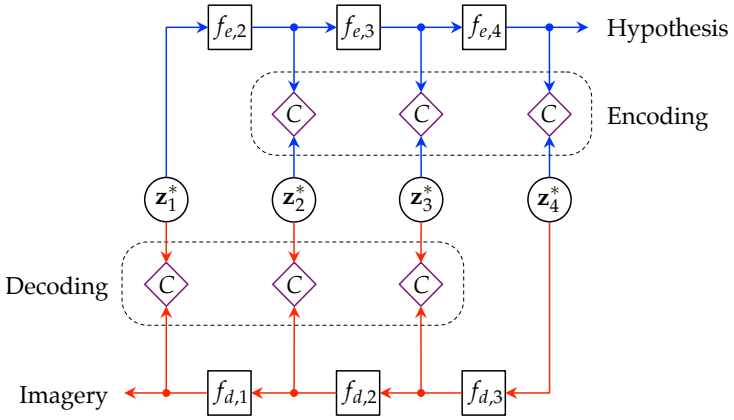
Fixing every $\mathbf{z}_l$ as the optimal $\mathbf{z}_l^*$ found in the previous step, a one-step gradient descent on the parameters $\mathbf{W}_l$ and $\mathbf{V}_l$ can be performed for the entire network. For the set of parameters between layers l and $l+1$, the gradients can be derived as follows:

$$\frac{\partial \mathcal{L}_{net}^{(2)}}{\partial \mathbf{W}_l} \propto \sum_{m=l+1}^{L} (\mathbf{z}_m - \hat{\mathbf{z}}_{m,1}) \frac{\partial \hat{\mathbf{z}}_{m,1}}{\partial \mathbf{W}_l}, \tag{4.22}$$

$$\frac{\partial \mathcal{L}_{net}^{(2)}}{\partial \mathbf{V}_{l+1}} \propto \sum_{m=1}^{l} (\mathbf{z}_m - \hat{\mathbf{z}}_{m,L}) \frac{\partial \hat{\mathbf{z}}_{m,L}}{\partial \mathbf{V}_{l+1}}. \tag{4.23}$$



(a) **Step 1:** Coding optimization.



(b) **Step 2:** Dictionary learning.

Figure 4.7: Decomposed factor graphs for alternating optimization. (a) Step 1 focuses on optimizing the codes $\mathbf{z}_l$ based on prediction, and reconstruction costs. (b) Step 2 transfers the optimal codes to the parameters $\mathbf{W}_l$ and $\mathbf{V}_l$, which will be used for the encoding or decoding task.

Algorithm 4.3: SPREAD optimization

Initialize $\{\mathbf{W}_l : 1 \leq l < L\}$ and $\{\mathbf{V}_l : 1 < l \leq L\}$
repeat
  Get random training example $(\mathbf{x}, \mathbf{y})$
  Fix $\{\mathbf{W}_l : 1 \leq l < L\}$ and $\{\mathbf{V}_l : 1 < l \leq L\}$ `/* Code optimization */`
  repeat
    Computer $\mathcal{L}_{net}^{(1)}$ `// Equation 4.19`
    Update $\{\mathbf{z}_l : 1 \leq l \leq L\}$ `// Equation 4.20`
  until *convergence*
  Fix optimal codes $\{\mathbf{z}_l^* : 1 \leq l \leq L\}$ `/* Dictionary learning */`
  Computer $\mathcal{L}_{net}^{(2)}$ `// Equation 4.21`
  Update $\{\mathbf{W}_l : 1 \leq l < L\}$ `// Equation 4.22`
  Update $\{\mathbf{V}_l : 1 < l \leq L\}$ `// Equation 4.23`
until *convergence*

Overall deep learning strategy. Algorithm 4.3 describes the entire process of this two-step optimization that alternates between code optimization and dictionary learning. Similar to the method presented in the previous section, this optimization can be embedded within a three-phase deep learning strategy. The first phase performs greedy unsupervised learning by stacking encoder-decoder networks from the bottom up. The second phase performs SPREAD optimization. Finally, error backpropagation can be used to enhance discriminative classification performances.

4.5 Evaluation: Handwritten digit recognition

Experimental dataset. The MNIST dataset of handwritten digits [LeCun et al., 1998] was used for performance evaluation. As previously described in Section 3.4.2, the dataset contains images of handwritten digits of 28×28 pixels each (Figure 3.11). The task is to classify an image into one of ten digits. The dataset is split into $60,000$ training images and $10,000$ test images. Many different methods have used this dataset as a benchmark. The basic version of this dataset, with neither preprocessing nor enhancements, was used for the evaluation. The error rate on the test set (percentage of wrong classifications) is the default metric for performance evaluation.

General experimental setup. The network was setup to be isomorphic to the DBN evaluated by Hinton et al. [2006]. It consists of five layers, inclusive of the input and output layers, as shown in Figure 4.8. Consecutive layers are linked by fully-connected weights. The first layer takes inputs from a 28×28 image through a 784-dimensional vector. The next two layers have 500 latent variables each and the third latent layer forms a 2000-dimensional representation. This large vector is associated with a one-hot coded output vector corresponding to 10 class labels, each representing a digit.

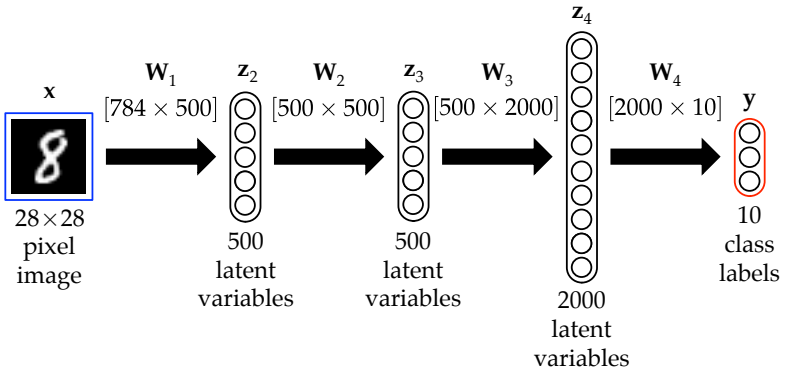


FIGURE 4.8: Deep network for handwritten digit recognition. This five-layer (784-500-500-2000-10) structure follows that of Hinton et al. [2006].

Training and testing procedure. The evaluation was performed using the two proposed architectures:

1. Top-down regularized deep belief network (Section 4.3), and
2. Predictive and reconstructive encoder-decoders (Section 4.4).

The three-phase learning strategy was adopted for training the networks. The following procedure was employed for training and testing the network. In addition, results using only phase 2 were also reported.

Phase 1. In the first phase, the networks were trained layer-by-layer using a subset of 44,000 training images, split into 440 mini-batches, with each having 10 examples of each class making up 100 images per mini-batch. The weight parameters for each shallow network (RBMs or encoder-decoder network) were updated after each batch.

Phase 2. The same set of data (44,000 images) was used for the second training phase. For model selection, a validation set of 10,000 images from the remainder of the training set was used. Deep learning (forward-backward algorithm for the DBN and SPREAD learning algorithm for deep encoder-decoders) was performed for 300 sweeps through the training data. Using the best performing model for each setup, the parameters were further updated by training it on all 60,000 images of the training set, split into 600 mini-batches of 100 images per batch. Up to this juncture, the evaluation protocol of the DBN of Hinton et al. [2006] was followed precisely.

Phase 3. From the resulting model, the whole network was fine-tuned using the error backpropagating algorithm in the third phase. All parameters were subsequently refined based on the gradient of a discriminative loss function. The training set was split into a 50,000-image training set and a 10,000-image validation set for parameter selection. Again, the full training set of 60,000 images was used to perform the final tuning for the model.

4.5.1 Results: top-down regularized deep belief network

To objectively analyze the effects of the top-down regularization, three architectural setups were evaluated:

1. Stacked RBMs with forward-backward learning,
2. Stacked sparse and selective RBMs with forward-backward learning, and
3. Forward-backward deep learning from randomly initialized weights.

Stacked RBMs with forward-backward learning. In the first phase, a stack of three RBMs were greedily trained: $\{\mathbf{W}_1 \in \mathbb{R}^{784\times500}, \mathbf{W}_2 \in \mathbb{R}^{500\times500}, \hat{\mathbf{W}}_3 \in \mathbb{R}^{510\times2000}\}$, in the manner described by Hinton et al. [2006], which gave a score of 2.49%. $\hat{\mathbf{W}}_3$ was then split into two parameter sets: $\mathbf{W}_3 \in \mathbb{R}^{500\times2000}$ and $\mathbf{W}_4 \in \mathbb{R}^{2000\times10}$, where $\mathbf{W}_4$ corresponds to the parameters for the output vector. For the second phase, instead of performing up-down learning, the forward-backward learning was performed, as described in Section 4.3 (Algorithm 4.1). After fine-tuning the parameters using the forward-backward algorithm, the error rate reduced to 1.14%. As a direct comparison, the DBN had a reported error rate of 1.25% after the up-down algorithm [Hinton et al., 2006]. Finally, a third phase discriminatively optimized the parameters of the entire network, bringing the error rate down to 0.98%.

Stacked sparse and selective RBMs with forward-backward learning. The RBMs were regularized using the point- and instance-wise regularization method as proposed in Section 3.3.1. A jointly sparse and selective regularizer was used as the target (see Section 3.3.2). After greedy training and learning the final RBM, the error rate of 2.14% was achieved. This method outperforms the conventional RBM stack, enhancing the importance of performing regularization during RBM learning. After the second phase of forward-backward learning, the error rate was brought down to 1.06%. Eventually, error backpropagation reduced the error rate to 0.91%. This is the best performing model among those evaluated. Figure 4.9 shows the 91 wrongly classified test examples.

FIGURE 4.9: Wrongly classified test examples by the top-down regularized deep belief network. There were 91 classification errors, many with challenging appearances.

Forward-backward deep learning from randomly initialized weights. As an exercise, the parameter initialization by phase 1 was not performed. Instead, a stack of RBMs with random weights was used. After training the model using the forward-backward learning, the classification score was 1.61%, which is fairly decent, considering that the network was optimized globally from scratch.

The results of the three evaluation setups are summarized in Table 4.4. As a comparison, the results of Hinton et al. [2006] are also reported.

Table 4.4: Results on MNIST after various training phases.

Learning algorithm		Classification error rate		
Phase 1 (stacking)	Phase 2 (fine-tuning)	Phase 1	Phase 2	Phase 3*
Deep belief network reported by Hinton et al. [2006]				
RBMs	Up-down	2.49%	1.25%	–
Top-down regularized deep belief network (Section 4.3)				
RBMs	Forward-backward	2.49%	1.14%	0.98%
Sparse & selective RBMs	Forward-backward	2.14%	1.06%	0.91%
Random weights	Forward-backward	–	1.61%	–
Predictive and reconstructive encoder-decoders (Section 4.4)				
Encoder-decoders	SPREAD	2.67%	1.25%	1.03%
Random weights	SPREAD	–	1.58%	–

*Phase 3 runs the error backpropagation algorithm whenever employed.

4.5.2 Results: predictive and reconstructive encoder-decoders

For this part of the evaluation, a deep network of predictive and reconstructive encoder-decoders described in Section 4.4 was used. Two experimental setups were evaluated:

1. Stacked encoder-decoders with SPREAD optimization, and
2. SPREAD optimization from randomly initialized weights.

Their classification scores on the MNIST test set are also presented in Table 4.4.

Stacked encoder-decoders with SPREAD optimization. A stack of sparse and selective encoder-decoder networks was first learned. Each new layer learns the coding and their projection for reconstructing the previous layer. The first three encoder-decoder networks $\{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3\}$ were learned with unsupervised learning. A final layer of parameters $\mathbf{W}_4$ was learned as a perceptron to map the 2000-dimensional vector to the class labels $\mathbf{y}$. This layer can be seen as learning the encoder that projects to the optimal coding, given by $\mathbf{y}$ in this case. This network produced an error rate of 2.67% after training with cross-validation. Subsequently, the SPREAD optimization described in Section 4.4 (Algorithm 4.3) performs an iterative alternating optimization between the coding and the weights (or dictionary), resulting in a score of 1.25%, similar to the DBN [Hinton et al., 2006]. Finally, error backpropagation is used to enhance discriminative classification performances, shrinking the classification rate to 1.03%.

SPREAD optimization from randomly initialized weights. Since the SPREAD optimization performs network-wide learning, it can operate as a standalone learning algorithm. The stack of encoder-decoders was merely initialized with random weights

and not learned. With this global optimization model, the error rate obtained was 1.58%, which is commendable score.

Figure 4.10 shows the deep network producing an imagery of each of the 10 classes by performing the decoding task $\hat{\mathbf{z}}_{1,L}$ from individual output units in the top layer $\mathbf{z}_L$ to the input layer $\mathbf{z}_1$. The imagery is coherent with the expected appearance of the digits. Figure 4.11 shows the state of the deep network in response to a training example after optimizing the code in all layers in the first step. The reconstructions and predictions from the optimized code for each layer is shown. As we examine deeper into the network, class prediction becomes more accurate and confident. Meanwhile, the representation becomes more and more invariant and the reconstruction ability suffers. It is interesting to see the gradual transition of the reconstruction from being instance-bound in the lower layers to model-based in the upper layers.

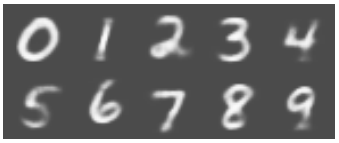


FIGURE 4.10: Imagery created from output unit decoding. Decoding from each of the 10 output units to the first layer produces a class-specific imagery with the appearance of a digit.

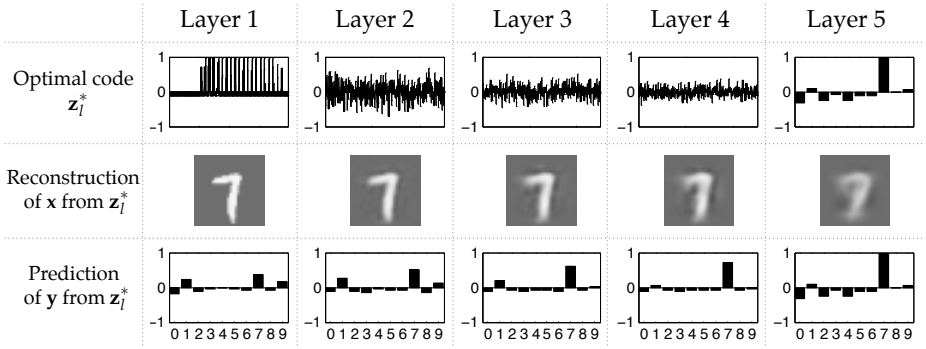


FIGURE 4.11: Reconstructions $\hat{\mathbf{z}}_{1,l}$ and predictions $\hat{\mathbf{z}}_{L,1}$ from optimized codes $\mathbf{z}_l^*$. The first row shows the optimized coding $\mathbf{z}_l^*$. The second and third row show the input reconstructions and output predictions respectively, when activated from the optimized code in the corresponding layer.

4.5.3 Summary of results

The top-down regularized DBN produced the best scores of 0.91% in the evaluation. The best results for the encoder-decoder-based deep network was 1.03%. These were significantly better than the 1.25% error rate of the DBN [Hinton et al., 2006]. Each of the three learning phases helped to improve the overall performance of the networks. Particularly, the algorithms employed in phase two – the forward-backward learning and the SPREAD optimization – were effective as a standalone algorithm, demonstrating its potential by learning from randomly initialized weights.

Overall, the results achieved are very competitive for methods with the same complexity that rely on neither convolution nor image distortions and normalization. A variant of the deep belief net [Salakhutdinov and Hinton, 2007], which focused on learning nonlinear transformations of the feature space for nearest neighbor classification, had an error rate or 1.0%. The deep convex net [Deng and Yu, 2011], which utilized more complex convex-optimized modules as building blocks but did not perform fine-tuning on a global network level, got a score of 0.83%. At the time of writing, the best performing model on the dataset gave an error rate of 0.23% and used a heavy architecture of a committee of 35 deep convolutional neural nets with elastic distortions and image normalization [Cireşan et al., 2012].

4.6 Summary and discussion

This chapter proposed the notion of deep learning by gradually transitioning from being fully-unsupervised to strongly-supervised. This is achieved through the introduction of an intermediate phase between the unsupervised and supervised learning phases. This notion is implemented as two methods to perform global supervised optimization of a deep network. The first method focuses on incorporating top-down information to each RBM of a DBN through regularization. The method is easily integrated into the intermediate learning phase based on a simple building block directly adapted from the regularization algorithm introduced in Chapter 3. The second method optimizes encoder-decoder networks by learning the projections to optimal codings that perform input reconstruction and output prediction. The methods can be performed as part of a three-phase learning strategy after greedy layer-wise unsupervised learning and before discriminative fine-tuning using error backpropagation. Empirical evaluations show that the methods lead to competitive results for handwritten digit recognition.

The same three-phase learning strategy was used to perform visual dictionary learning in Chapter 5, whereby the first phase employed the sparse and selective regularization proposed in Chapter 3 and the dictionary was fine-tuned using top-down regularization using the forward-backward learning algorithm introduced in Section 4.3.2 and being re-optimized through error backpropagation.

5 Learning Hierarchical Visual Codes

Chapter abstract — The four steps of the bag-of-words model are namely, 1) the extraction of local descriptors, 2) local feature coding via a learned visual dictionary, 3) pooling and 4) classification. This chapter proposes to tackle the problem of feature coding by learning a hierarchical visual dictionary that maps the local descriptors into richer mid-level representations. This work builds upon the proposed learning algorithms for regularized unsupervised learning and deep supervised learning techniques, introduced in Chapters 3 and 4 respectively. The resulting visual codes from this step will be the starting point for optimizing the subsequent pooling step (Chapter 6).

The work in this chapter has led to the publication of a journal article and a conference paper, as follows:

- Goh, H., Thome, N., Cord, M., and Lim, J.-H. (2013). Learning deep hierarchical visual feature coding. *IEEE Transactions on Neural Networks and Learning Systems*. [Goh et al., 2014]
- Goh, H., Thome, N., Cord, M., and Lim, J.-H. (2012). Unsupervised and supervised visual codes with restricted Boltzmann machines. In *European Conference on Computer Vision (ECCV)*. [Goh et al., 2012]

5.1 Introduction

Visual recognition is one of the basic and trivial aspects of human perception, but remains a major challenge in artificial intelligence. The problem arises when trying to understand complex semantic concepts, such as objects and scenes, from raw digitalized image pixels. Visual information processing has been typically treated as a hierarchical bottom-up pipeline in which visual representations are processed sequentially with increasing complexity, from [Marr, 1976]. To tackle the computer vision problem, the bag-of-words (BoW) model (Figure 5.1), performs a sequence of data processing steps – feature extraction, feature coding, pooling and classification – to take the visual representation from the low-level pixels to the high-level semantic representations for classification to take place. The model has been comprehensively reviewed in Section 2.4.

Chapter 5
Learning hierarchical visual codes

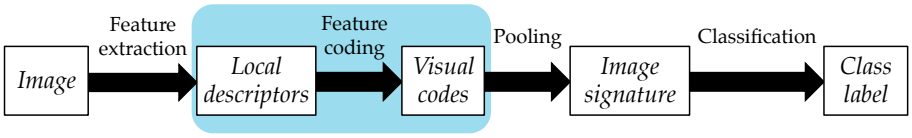


Figure 5.1: The feature coding step within the bag-of-words pipeline

In the BoW model, visual processing begins with the extraction of local image descriptors (see Section 2.4.2), such as the scale invariant feature transform (SIFT) [Lowe, 1999], the histogram of oriented gradients (HOG) [Dalal and Triggs, 2005] and the speeded-up robust features (SURF) [Bay et al., 2008]. SIFT descriptors form a robust and powerful representation of the local image appearance and is the starting point for further abstraction of the visual representation. In this thesis, the SIFT descriptors are sampled from an image using a dense and overlapping uniform grid.

At the top-end of the visual processing pipeline, classification is performed (Section 2.4.5). The BoW model employs a classification model, such as support vector machines (SVM), to predict the class label from an image signature – a single high-dimensional vector describing the appearance of the entire image. This vector is generated through a pooling step that uses spatial pyramids [Lazebnik et al., 2006] to aggregate descriptors across spatial partitions into histograms (Section 2.4.4). It was shown that using max-pooling to generate the signature tend to work well, especially when linear classifiers are used [Boureau et al., 2010a].

The use of SIFT descriptors, spatial pyramidal max-pooling and SVM classifiers have been crucial in producing state-of-the-art image classification results for BoW models. Although the local descriptors are more abstract than image pixels, a representational gap still exists between the image descriptors and higher-level representations used for classification. To fill this gap, a feature coding step maps each descriptor to an intermediate mid-level representation of visual codes (Section 2.4.3). Exploiting machine learning techniques to learn a visual dictionary for feature coding is currently the subject of much research. This chapter specifically focuses on using deep learning to learn hierarchical visual dictionaries.

5.2 Single-layer feature coding

Restricted Boltzmann machines (RBM) [Smolensky, 1986] have been effective in modeling input data using a layer of latent representations (see a review in Section 2.2.3). In the rest of this section, I will describe my method to learn a visual dictionary and perform feature coding using a regularized RBM (Section 3.3.1).

5.2.1 Unsupervised visual dictionary learning

In the context of feature coding, the RBM is a two-layer network that represents a local image descriptor $\mathbf{x}$ in its input layer and maps it to a visual code represented by its latent layer $\mathbf{z}$ (see Figure 5.2). The input layer contains I dimensions corresponding to the size of a local descriptor vector (i.e. 128 dimensions for SIFT). The latent layer consists of J latent variables, each being the response to a visual word in the dictionary. Offset units, x_0 and z_0 are also added and permanently set to one. The descriptor and the visual code are connected by an undirected weight matrix $\mathbf{W}$, such that every input unit i is connected to every codeword j via a weighted connection w_{ij}. $\mathbf{W}$ also serves as the visual dictionary containing J visual words $\mathbf{w}_j \in \mathbb{R}^I$.

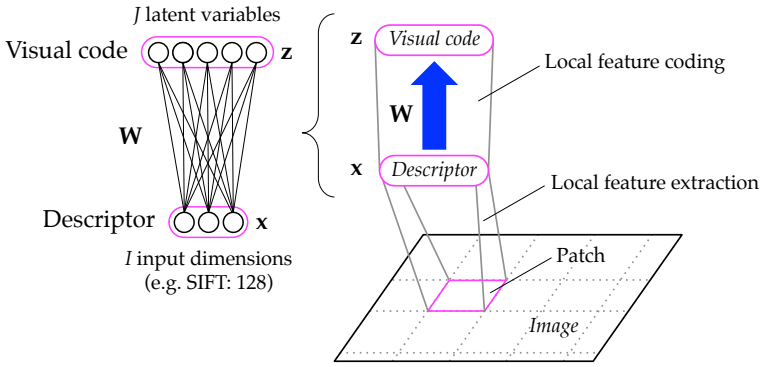


FIGURE 5.2: Encoding a local descriptor with a restricted Boltzmann machine. The input layer takes in a local descriptor and the latent layer forms the visual code representation for that descriptor.

As shown in Figure 5.2, the feature coding operation considers each local descriptor to be spatially independent. The weights are shared across the image at every spatial position defined by a tiling operation. Given a local descriptor x and the dictionary $\mathbf{W}$ the visual code z_j can be computed by a feedforward activation:

$$z_j = f_e(\mathbf{x}, \mathbf{W}) = \frac{1}{1 + \exp(-\mathbf{W}^T\mathbf{x})}. \tag{5.1}$$

This function applies a point-wise nonlinearity through the logistic function to the result of a linear transformation. When using a binary RBM, normalization may be required to be performed to parse the local descriptor as an input. For SIFT descriptors, the vectors are first ℓ_1-normalized so that each sums to a maximum of one. This results in a quasi-binary input representation to suit the binary RBM.

An important advantage of using an RBM for feature coding over the more commonly used decoder networks, such as those of Yang et al. [2009], Wang et al. [2010] and Boureau et al. [2010a, 2011], is inference speed. Unlike decoder networks, RBMs do not need to perform any optimization during feature coding. Given a local descriptor, the simple feedforward mapping results in a fast generation of the visual code.

The RBM uses a learning algorithm known as contrastive divergence to learn suitable mapping parameters **W**. The optimization attempts to maximize the likelihood of the input data distribution by performing gradient descent while iterating over a training set of local descriptors (Section 2.2.3). Through the latent layer, the RBM models the statistics of the interactions between various dimensions of the local descriptor. The process is entirely unsupervised and does not require the use of any information of the image class.

However, using input maximum likelihood as the only criteria to discover the mid-level representations may not be suitable for the image classification task. To learn sensible visual codes for image classification, regularization may be incorporated to guide the learning of the structure of the data (Section 3.3.1). To model local descriptors, the sparse and selective distribution-based prior is used to bias the RBM learning. This was comprehensively elaborated in Section 3.3.2.

For image classification, selectivity and sparsity in the feature coding may help improve visual information efficiency, memory storage capacity and, most importantly, pattern discrimination [Barlow, 1989; Rolls and Treves, 1990]. As defined in Section 3.2.1, sparsity is a summary of the element in the *visual code vector* of one local descriptor, while selectivity is a statistic of *one latent variable* across a set of local descriptor instances. It is important for a representation to be jointly sparse and selective. A simplified graphical explanation of this is shown in Figure 5.3, whereby feature discrimination suffers when the coding is not jointly sparse and selective. With suitable regularization performed during learning, interesting spatially coherent representations can be learned from local gradient-based descriptors (see Section 5.5.2).

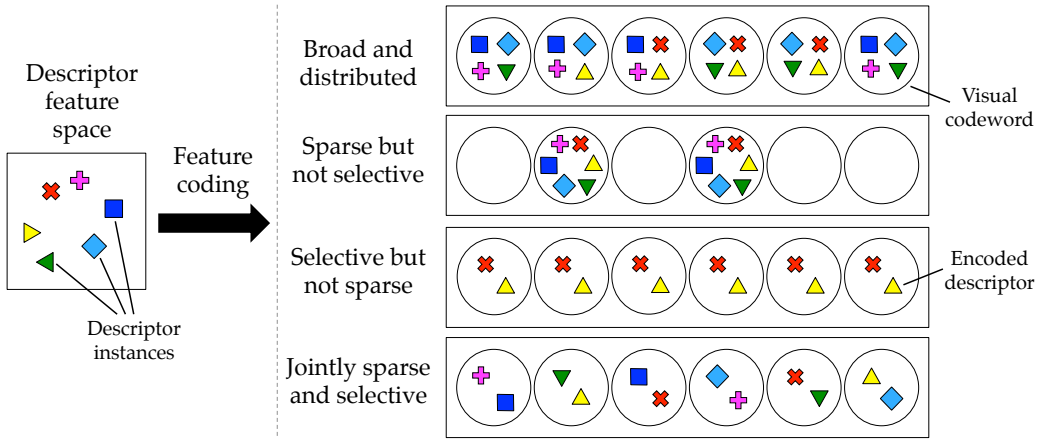


Figure 5.3: Importance of a jointly sparse and selective visual coding. Descriptors encoded using broad and distributed coding have highly overlapping representations and lack discriminative power. Having only sparsity in the coding alone may lead to visual codewords that do not specialize. A coding with only selectivity may cause redundant codewords that memorize only a small subset of features. Selectivity and sparsity should jointly exist to form meaningful and discriminative visual codes.

Encoding macro features. Each local descriptor independently represents the pixel information from its spatial window, without considering its neighbors. To enhance the richness in the representation and model descriptor correlations in the image space, it may be advantageous to incorporate spatial information into the feature coding structure, instead of treating the descriptors as being spatially independent. In this context, a more complex image descriptor, known as the *macro feature* [Boureau et al., 2010a], can be extracted from the image. A macro feature concatenates local descriptors over a spatial neighborhood and a RBM can learn a visual dictionary that models the interactions, not only within a descriptor but also between neighboring ones. Figure 5.4 shows how feature coding can be performed using macro features as inputs.

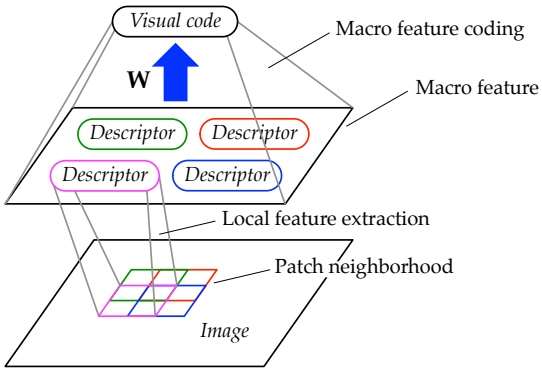


FIGURE 5.4: Macro feature extraction. The visual code of a macro feature captures both inter- and intra-descriptor correlations within a spatial neighborhood.

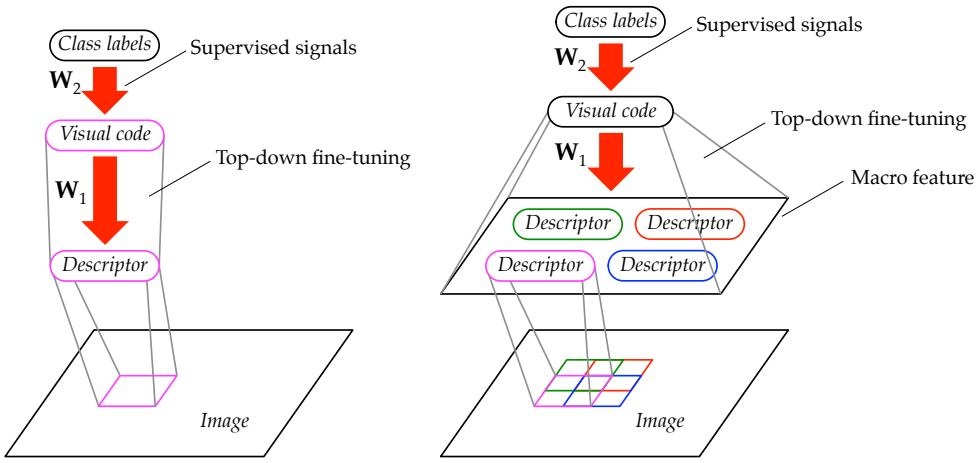
5.2.2 Supervised fine-tuning of single-layer visual dictionaries

Unsupervised learning models the structure of local descriptors through an optimization criterion, such as maximum likelihood estimators (Section 2.2.3) or minimizing input reconstruction costs (Section 2.2.2). However, as it is an appearance-based model learned solely from the bottom-up and any change in task or labeling for the same dataset is invisible to the unsupervised model. Without knowledge of the task, it is impossible to know if the learned visual representations is optimal for image classification, even when the most powerful unsupervised learning models are used. In this context, supervised dictionary learning is crucial for image classification and can complement unsupervised visual dictionary learning.

After training, the RBM visual dictionary can be used directly as a two-layer feedforward neural network. Due to this inherent structure, the parameters of the dictionary can be fine-tuned using top-down supervised learning. A classifier $\mathbf{W}_2$, which performs a linear projection followed by a softmax activation function can be added to learn the visual codes with the class labels. When combined with the RBM dictionary,

the architecture takes the form of a "shallow" multilayer perceptron network (Section 2.2.1). From this, the RBM visual dictionary $\mathbf{W}_1$ can be fine-tuned concurrently with the classification parameters $\mathbf{W}_2$ using the error backpropagation algorithm.

This way of connecting the class labels to the local visual codes deviates from classifying the entire image and only attempts to improve discrimination between local features independently. The expectation is that performance for the image classification will also be given a boost with more discriminative local representations. This is verified by empirical analysis in Section 5.5.3. Figure 5.5 shows how the supervised signals are incorporated into the fine-tuning of visual dictionaries for encoding local descriptors and macro features.



(a) From single local descriptors. (b) From macro feature.

Figure 5.5: Supervised fine-tuning of single-layer visual dictionaries. A classification layer is added on top of the representation of visual codes to enable the backpropagation of error signals to the locally applied visual dictionary.

5.3 Hierarchical feature coding

In this section, I propose a hierarchical feature coding scheme that extends the single-layer feature coding method presented in the previous section. The visual dictionary is learned using the proposed deep learning techniques by combining sparse and selective RBMs proposed in 3.3 with the top-down deep regularization presented in 4.3. To the best of my knowledge, this is the first attempt on learning deep visual representations from local descriptors rather than pixels. From the perspective of the BoW model, it is the first feature coding scheme that uses a deep visual dictionary. The proposed deep feature coding method outperforms existing shallow local descriptor feature coding methods and deep pixel-based architectures (see Section 5.5.1).

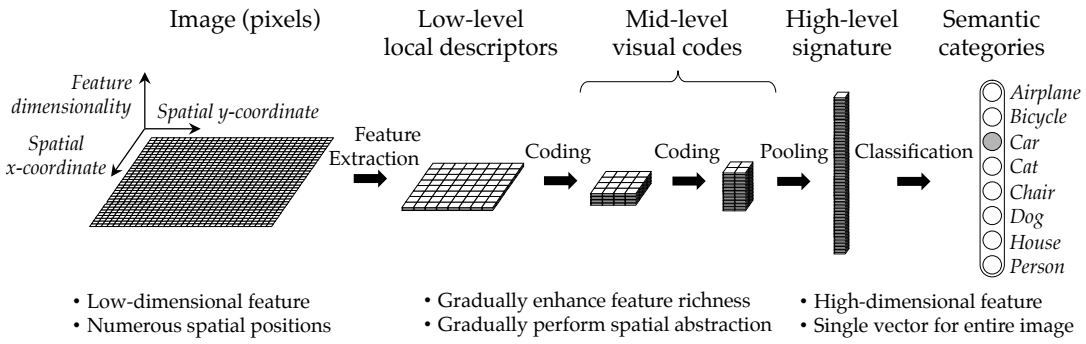
5.3.1 Motivations of hierarchical feature coding

The general motivations of learning deep architectures have been covered in Section 2.3.1. Specific to computer vision, existing deep architectures for image modeling learn visual representations from pixels [Lee et al., 2009; Kavukcuoglu et al., 2009; Yu et al., 2011; Krizhevsky et al., 2012]. While these attempts at learning from pixels have resulted in the discovery of interesting features, such as Garbor-like filters [Lee et al., 2008], in general, image classification performances remain below those of the BoW model. The proposed deep feature coding scheme learns a hierarchical visual dictionary from local descriptors as a starting point with a greater representational power than raw pixels. This hybridization integrates the visual modeling capabilities of local descriptors and spatial pyramidal pooling of the BoW model with the adaptability and representational power of deep learning.

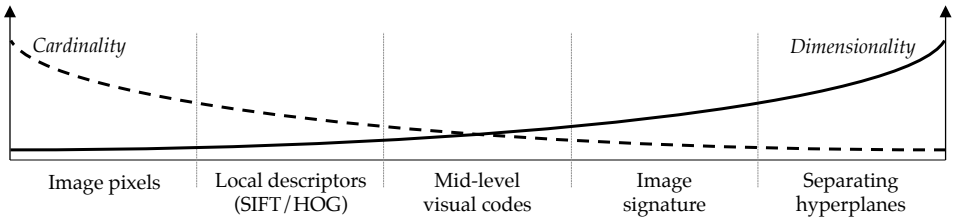
The BoW model transforms image representations from pixels to low-level descriptors, mid-level visual codes, and eventually to a high-level image signature, which is used for classification. The depth of a feature coding scheme allows for a gradual mapping of representations to achieve both feature enhancement and abstraction during the coding process. The scheme continues the general tactic of the BoW model of progressively increasing dimensionality while reducing cardinality in every successive step (Figure 5.6).

5.3.2 Stacking and spatially aggregating visual dictionaries

In deep belief networks [Hinton et al., 2006], layers of RBMs are stacked and trained in a greedy manner. Each pair of adjacent layers is fully-connected to each other. However, this conventional method of stacking fully-connected layers does not exploit spatial information and may not scale well to larger images. When representing visual information with a hierarchy, an intuitive plan is to first represent parts of objects in the lower levels and combine these parts into whole objects and scenes in the higher levels. Convolutional neural networks [LeCun et al., 1989, 1998; Lee et al., 2009] and biologically inspired models [Riesenhuber and Poggio, 1999; Bileschi et al., 2007; Mutch and Lowe, 2008; Theriault et al., 2013b] model this expanded spatial dimensionality through the use of spatial sub-sampling operations. With every new layer the complexity of the features increases and the areas of representation over the original image are enlarged. When learning hierarchical visual dictionaries, it is sensible to perform spatial aggregation, whereby higher-level concepts are formed by abstracting over the lower-level ones.



(a) Bag-of-words model with hierarchical feature coding.



(b) Relation between representational dimensionality and cardinality through the framework.

Figure 5.6: Representation transformation by the bag-of-words (BoW) model. The image representation is transformed from image pixels to low-level descriptors, then to mid-level visual codes, and eventually to a high-level image signature, which is used for classification. (a) During hierarchical feature coding, the feature dimensionality increases progressively, enhancing the richness of the representation. Meanwhile, the features are gradually abstracted through the layers and the number of features (i.e. cardinality) over the image space is reduced. (b) With every operation, dimensionality increases, while cardinality decreases.

Figure 5.7 shows the connections between a stack of two RBMs, whereby the higher-level RBM spatially aggregates visual codes of the lower-level one. The lower-level RBM $\mathbf{W}_1$ is first trained as a single unsupervised feature coding layer to produce visual codes $\mathbf{z}_2$ (see Section 5.2). In a similar manner to how macro features are concatenated, the visual codes $\mathbf{z}_2$ within the same neighborhood are combined into a more complex feature. For the purpose of consistency, this aggregated visual code shall be known as a *macro code*. A second RBM $\mathbf{W}_2$ is stacked above the first RBM. This higher-level RBM is trained to encode the macro code into a high-level visual code $\mathbf{z}_3$. Due to the spatial concatenation, the interactions between the low-level visual codes in the same neighborhood will be modeled by the higher-level RBM.

Extending from their single-layer counterparts shown in Figure 5.2 and Figure 5.4, the two-layer visual dictionaries for encoding local descriptors and macro features are shown in Figure 5.8(a) and Figure 5.8(b) respectively. When using a hierarchy to encode a macro feature, spatial aggregation is performed twice – once in each new RBM layer.

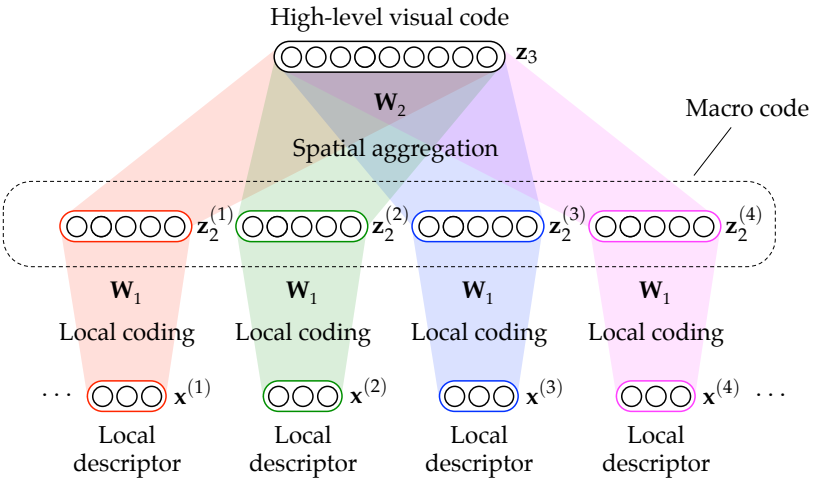
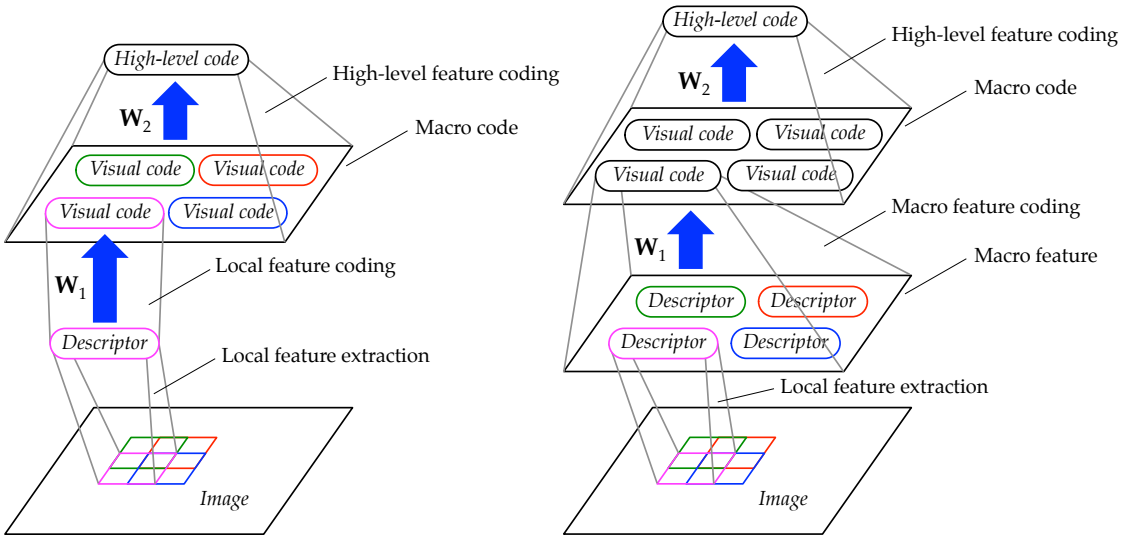


FIGURE 5.7: Spatial aggregation by the higher-level visual dictionary. The higher-level visual dictionary $\mathbf{W}_2$ models the interactions between local visual codes encoded at different locations by the lower-level visual dictionary $\mathbf{W}_1$.



(a) From single local descriptors.

(b) From macro feature descriptors.

FIGURE 5.8: Two-layer hierarchical feature coding. The hierarchy is formed by stacking a second RBM $\mathbf{W}_2$ on top of existing single-layer visual dictionaries $\mathbf{W}_1$ for encoding (a) local descriptors and (b) macro features. The higher-level RBM spatially aggregates visual codes from the lower level RBM.

5.3.3 Top-down regularization of hierarchical visual dictionaries

So far, the training algorithms have focused on unsupervised learning. Yet, just like in the case of the single-layer visual dictionary, it is important for supervised learning to be introduced for the hierarchical visual dictionary. However, due to the depth of the architecture, more care has to be taken to integrate supervision into the learning.

Using top-down discrimination to model the data is an effective way to approach a classification task. In a deep architecture, the discriminative signal diffuses as it travels

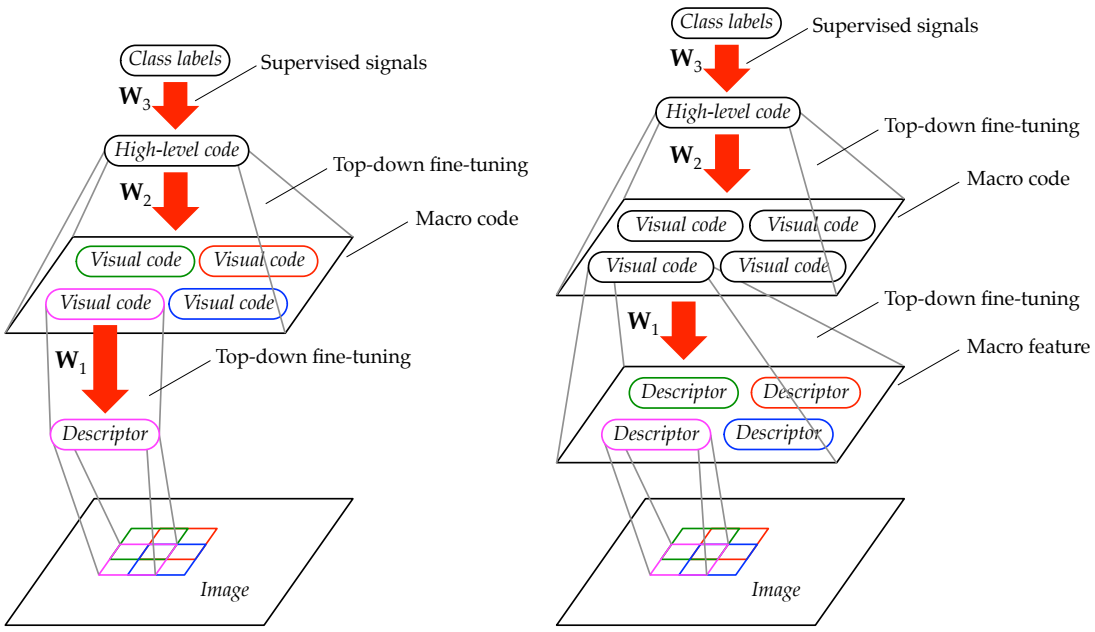
down the network (see Section 2.2.1). Similarly, the importance of supervised learning increases as we stack layers, since the quality of the unsupervised representations diminishes at the upper layers (Table 5.2). A combination of both bottom-up and top-down learning needs to be performed to train an effective model. In this architecture, after greedily stacking the RBMs using sparsity and selectivity (Section 3.3), class labels are introduced through two supervised fine-tuning phases: top-down regularization and discriminative error backpropagation.

Top-down regularization. The learning strategy follows the forward-backward learning algorithm proposed in Section 4.3.2. This iterative algorithm can be dissected into the following sub-steps.

- *Initialization.* A new "classifier" RBM, with weights $\mathbf{W}_3$, connects the high-level visual codes $\mathbf{z}_3$ (see Figure 5.7) to a softmax activated output layer $\mathbf{y} \in \mathbb{R}^C$, with each unit corresponding to the class label c. This RBM is trained by directly associating the $\mathbf{z}_3$ to outputs $\mathbf{y}$ and can be updated via gradient descent using Equation 4.7 with Equation 4.10 to update the parameters of this classifier.
- *Forward-backward learning.* A initial forward pass generates the visual codes $\mathbf{z}_{l,l-1}$ from the bottom-up using Equation 4.4. A backward pass then samples target visual codes $\mathbf{z}_{l,l+1}$ downwards from the higher-level layer with Equation 4.11. Equation 4.7 can then be used to fine-tune the two dictionaries and the local classifier.

Discriminative error backpropagation. After top-down regularization, discriminative fine-tuning is performed. Various discriminative loss functions could be used to update the RBMs' parameters, with the most standard being the classification softmax cross-entropy loss leading gradient descent updates using the error backpropagation algorithm. This is nicely adapted for multi-layered neural networks. This fine-tuning is also performed in the single-layer visual dictionary described in Section 5.2.2.

Figure 5.9 shows the supervised signals being incorporated into the fine-tuning of the visual dictionary hierarchy for local descriptors and macro features. The supervised optimization for the deep visual hierarchy faces the same drawback as her single-layered cousin. The optimization is still performed over local regions of the image using a globally defined image label. However, since spatial aggregation is performed for this hierarchical model, the regions represented are larger than the single-layer model, especially when macro features are used.



(a) From single local descriptors.

(b) From macro feature.

FIGURE 5.9: Supervised fine-tuning of hierarchical visual dictionaries.

5.3.4 Three-phase training of the bag-of-words model

The training process that has been described, so far, has been for learning the hierarchical visual dictionary. Together with the training of the support vector machine for classification in the final step, the entire procedure of six training steps for the bag-of-words model can be summarized into three phases, mimicking the strategy proposed in Section 4.3.3. The result of one step is used to initialize the parameters for the next. Figure 5.10 shows then entire training procedure for the BoW model, as well as the inference process. During the inference process, a single feedforward pass through the BoW pipeline outputs the class label prediction of an image from the pixel-level.

1. *Phase 1 - greedy unsupervised learning.* RBM visual dictionaries are learned layer-by-layer from the bottom up. Learning is regularized with selectivity and sparsity in each layer (see Section 3.3). Spatial aggregation models the structure of the visual codes within spatial neighborhoods (see Section 5.3.2).

2. *Phase 2 - top-down regularized deep learning.* A top-level classifier is learned, while the visual dictionaries are fine-tuned using a forward-backward learning strategy, as described in the previous section.

3. *Phase 3 - supervised discriminative learning.* Discriminative learning is exploited to enhance the discriminative power of the local visual dictionaries through error backpropagation. Subsequently, the images in the training set can be encoded through the spatial hierarchy to generate the high-level visual codes. Spatial

pooling is then performed using spatial pyramids [Lazebnik et al., 2006] with max-pooling [Boureau et al., 2010a]. The result is a single vector – an image signature – that describes the entire image. The collection of image signatures together with their corresponding class labels are used to discriminatively train the SVM for the final classification step.

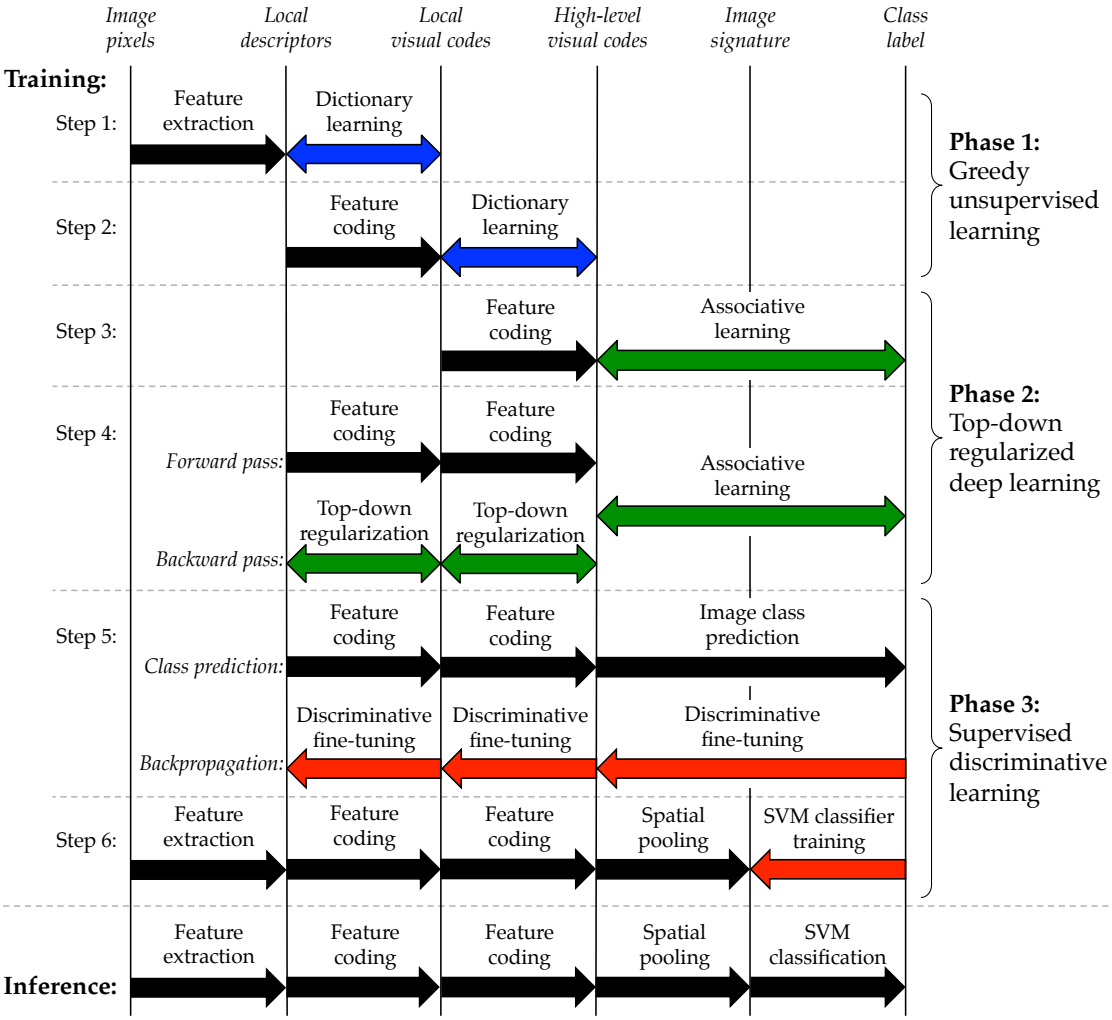


Figure 5.10: Training the bag-of-words model in six steps. The steps are grouped into three phases: 1) greedy supervised learning, 2) top-down regularized deeps learning and 3) supervised discriminative learning. Inference consists of a single feedforward pass through the bag-of-words pipeline.

5.4 Image classification experimental setups

Various object and scene classification tasks were used to evaluate the single-layer and hierarchical visual dictionaries and feature coding schemes presented. The performance of an image classification model can be empirically quantified by applying the model on a given experimental dataset. In this section, I will introduce the main evaluation metric employed and three image classification datasets used in this thesis.

5.4.1 Image classification datasets

The three datasets used to evaluate image classification performances are namely: 1) the 15-Scenes dataset [Lazebnik et al., 2006], 2) the Caltech-101 dataset [Fei-Fei et al., 2004] and 3) the Caltech-256 dataset [Griffin et al., 2007].

Fifteen Scene Categories. The fifteen scene categories (15-Scenes) dataset [Lazebnik et al., 2006] consists of 4,485 images from 15 different scene categories. The dataset is an expansion of the thirteen category dataset released by Fei-Fei and Perona [2005], which was extended from the dataset by Oliva and Torralba [2001]. Each image is assigned a category and there are between 210 to 410 images per category. Figure 5.11 shows an example image from each of the category. The objective of the dataset is to predict the category of an image. The dataset is publicly available[1] and is widely used within the computer vision community for evaluating image classification models.

Caltech-101. The Caltech-101 dataset [Fei-Fei et al., 2004] is one of the most popular datasets for evaluating image classification models. It contains 9,144 images belonging to 101 object categories and one background class. There are between 31 to 800 images in each category, with most categories containing about 50 object instances. Each image is roughly 300×200 pixels in size. The authors of this dataset also included carefully segmented object masks. However, this is not used for the purposes of this thesis. The dataset is available online[2]. Some examples of this dataset are shown in Figure 5.12. In general, the objects tend to be nicely centered within the image and most images have little or no clutter and object pose is also stereotypical [Ponce et al., 2006].

Caltech-256 The Caltech-256 dataset [Griffin et al., 2007] extends the original Caltech-101 dataset to 256 object classes and 30,608 images – a 3.35 times increase. Same as before, an extra background category is included to represent non-semantic clutter. The number of images per class also increased from a mean of 90 to a mean of 119 (see Table 5.1). In addition, 29 of the largest classes of the Caltech-101 dataset were retained in this dataset. Compared to the Caltech-101 dataset, the Caltech-256 dataset has more variations and the spatial alignment of the objects are more inconsistent. However, they still tend to be generally centered, as shown in Figure 5.13, which the authors attribute to photographer bias. This dataset is also available online[3].

[1] http://www-cvr.ai.uiuc.edu/ponce_grp/data
[2] http://www.vision.caltech.edu/Image_Datasets/Caltech101
[3] http://www.vision.caltech.edu/Image_Datasets/Caltech256

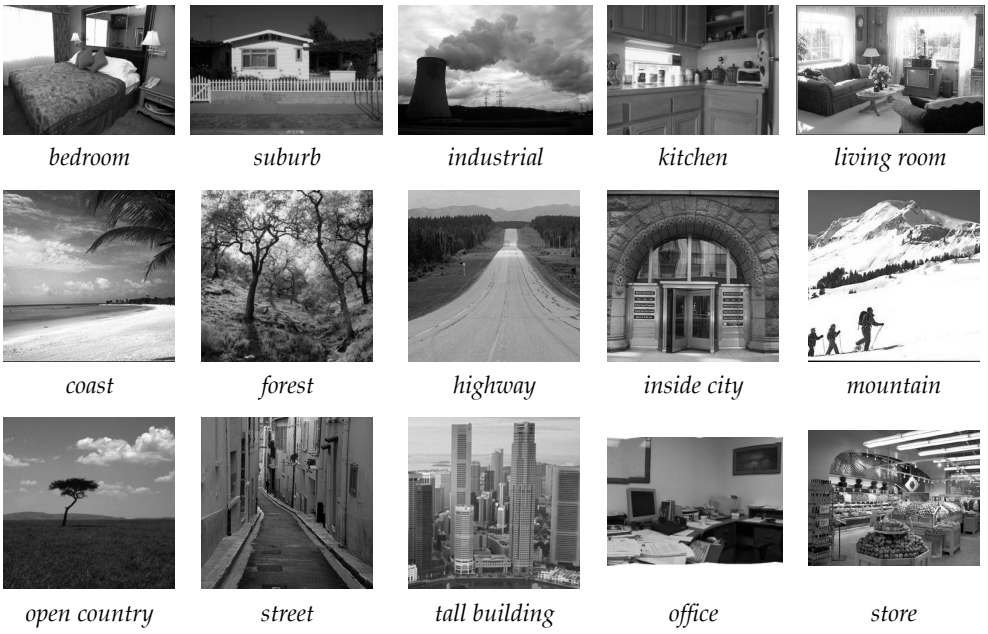


Figure 5.11: Example images from each scene category in the 15-Scenes Dataset.

Figure 5.12: Example images of objects from the Caltech-101 dataset.

Figure 5.13: Example images of objects from the Caltech-256 dataset.

5.4.2 Evaluation setup and metric

In a typical setup, for methods using unsupervised learning, a random set of input image descriptors is used to train the model. If supervised learning is employed, then a random number of images per class is randomly drawn from the dataset. Since neither of the three datasets have a predefined training or test set, there will be a need to perform several experimental trials with different train-test splits. Table 5.1 details the train-test splits for the three image classification datasets evaluated in this thesis.

TABLE 5.1: Train-test splits used for the three experimental datasets.

Dataset	Images per class	Training examples per class	
		Setting 1	Setting 2
15-Scenes	280	100 (33.4%)	
Caltech-101	90	15 (16.7%)	30 (33.5%)
Caltech-256	119	30 (25.1%)	60 (50.2%)

The images that are not used for supervised training are then placed into a test set for the purposes of a quantitative evaluation. From the test set, the image classification accuracies $\mathcal{A}_c$ are computed for each class c. The results are then averaged over the set of C classes to obtain the mean class-wise accuracy $\langle \mathcal{A} \rangle$. The evaluation is performed over T experimental trials t, with different randomly-selected set of images used for training and testing. The result is averaged over the T trials to obtain the final evaluation metric for image classification performance:

$$\mu_{\mathcal{A}} = \frac{1}{T} \sum_{t=1}^{T} \langle \mathcal{A} \rangle_t = \frac{1}{CT} \sum_{t=1}^{T} \sum_{c=1}^{C} \mathcal{A}_{c,t}. \tag{5.2}$$

The standard deviation across the trials is also reported:

$$\sigma_{\mathcal{A}} = \sqrt{\frac{1}{T} \sum_{t=1}^{T} (\langle \mathcal{A} \rangle_t - \mu_{\mathcal{A}})^2}. \tag{5.3}$$

The presentation of the result is typically formatted as "$\mu_{\mathcal{A}} \pm \sigma_{\mathcal{A}}$". For this thesis, a total of 10 trails were performed for the evaluation ($T = 10$).

5.4.3 Experimental setup

Visual dictionary setup and input descriptors. The main objective of the experiment was to evaluate and analyze the performance of different architectural setups. Single-layer and hierarchical visual dictionaries were used to encode two types of inputs – local image descriptors and macro features. The following four setups were evaluated.

- Single-layer dictionary from local descriptors (Figure 5.2 with Figure 5.5(a)).
- Single-layer dictionary from macro features (Figure 5.4 with Figure 5.5(a)).
- Hierarchical dictionary from local descriptors (Figure 5.8(a) with Figure 5.9(a)).
- Hierarchical dictionary from macro features (Figure 5.8(b) with Figure 5.9(a)).

Before feature extraction, images were resized while retaining their original aspect ratios, such that the longer spatial dimension was at most 300 pixels. SIFT descriptors [Lowe, 1999] were extracted from densely sampled patches of 16×16 at 8 pixel intervals. Macro features were concatenated from 2×2 neighborhoods of SIFTs extracted at 4 pixel intervals. This setup follows that of existing BoW approaches [Boureau et al., 2010a]. The SIFT descriptors were ℓ_1-normalized by constraining each descriptor vector to sum to a maximum of one.

Visual dictionary learning. A set of $200,000$ randomly selected descriptors were used as the training set for unsupervised dictionary learning. For supervised fine-tuning using the forward-backward learning and error backpropagation, a number of training images per class – 15 or 30 for Caltech-101; 30 or 60 for Caltech-256; 100 for 15-Scenes – were randomly drawn for each experimental trial (Section 5.4.2). The same train-test split was also used for the subsequent training the SVM classifier.

Pooling and classification. From the visual codes produced by each feature coding scheme, a three-level spatial pyramid [Lazebnik et al., 2006] with max-pooling [Boureau et al., 2010a] generated the final image signature based on the typical pooling grids of 4×4, 2×2 and 1×1. This vector of $21J$ elements, where J is the number of visual coding dimensions, was used to train a linear SVM to perform multi-class classification.

Evaluation metric. The evaluation metric is described in Section 5.4.2. The images that were not used for supervised training were then used as the test set. The mean $\mu_{\mathcal{A}}$ and standard deviation $\sigma_{\mathcal{A}}$ of the class-wise image classification accuracies over 10 trials were computed. For the Caltech-101 and Caltech-256 datasets, following the standard evaluation protocol, the evaluation metric was computed for using all class (including the background class) for both training and evaluation.

5.5 Image classification evaluation and discussions

5.5.1 Evaluation: image classification results

The image classification results on the Caltech-101 and 15-Scenes datasets using the different setups of the visual dictionary – with different depths and input descriptors – are presented at the bottom of Table 5.2. Also presented are the intermediate results after the unsupervised learning phase, whereby the SVM classifier is trained on the visual codes that are not fine-tuned with supervision. Essentially, the training process of the BoW model jumps from the results of phase 1 to step 6 directly (Figure 5.10). The best image categorization results obtained on the Caltech-101 dataset were $72.1 \pm 1.3\%$ and $79.7 \pm 0.9\%$, using 15 and 30 training images respectively. For the 15-Scenes dataset, a classification accuracy of $86.4 \pm 0.6\%$ was obtained.

The best performing model was consistently the hierarchical visual dictionary trained and fine-tuned with supervision on macro feature inputs. To the best of my knowledge, these are the best ever results for methods focusing squarely on feature coding in the BoW model. Moreover, performance was consistently good across both datasets. Macro features were observed to consistently outperform SIFT descriptors, by about 3% on the Caltech-101 dataset and 1.5% for the 15-Scenes dataset. This difference in performance validates the results reported by Boureau et al. [2010a].

The hierarchical visual dictionary also achieved competitive image classification performances on the Caltech-256 dataset, reported in Table 5.4. The model obtained average accuracies of 41.5 ± 0.7 and 47.2 ± 0.9 using 30 and 60 training examples respectively. As a reference, the results for other methods producing competitive results are shown at the bottom of Table 5.4. The Caltech-256 dataset was also used to analyze the properties of transfer learning in deep feature hierarchies (see Section 5.5.3).

Comparison with feature coding methods. Table 5.2 compares the results of the proposed feature coding schemes against other feature coding strategies that encode single feature types and follow the same BoW pipeline. The proposed method is favorably positioned among both the unsupervised and supervised methods. Compared against all other coding-based schemes, the leading results are obtained by the proposed hierarchical feature coding scheme, learned from macro features and fine-tuned with supervised learning. At this juncture, RBM-based methods for visual dictionary learning seem to gain a slight edge over other feature coding methods. RBM-based methods also have faster inference speeds as compared to decoder networks [Wang et al., 2010; Boureau et al., 2010a; Yang et al., 2009] and have a significantly more compact representation than some feature coding methods [Boureau et al., 2011].

Table 5.2: Performance comparison with other feature coding methods using the bag-of-words model.

Method	Authors	Dictionary Size	Caltech-101		15-Scenes
			15 tr. img.	30 tr. img.	100 tr. img.
Non-learned feature coding					
Hard assignment	Lazebnik et al. [2006]	200	56.4	64.6 ± 0.8	81.1 ± 0.3
Kernel codebooks	Van Gemert et al. [2010]	200	-	64.1 ± 1.5	76.7 ± 0.4
Soft assignment	Liu et al. [2011]	1000	-	74.2 ± 0.8	82.7 ± 0.4
Feature coding with decoder network					
ScSPM	Yang et al. [2009]	1024	67.0 ± 0.5	73.2 ± 0.5	80.3 ± 0.9
LLC	Wang et al. [2010]	2048	65.4	73.4	-
Sparse coding & max-pooling	Boureau et al. [2010a]	1024	-	75.7 ± 1.1	84.3 ± 0.5
Multi-way local pooling	Boureau et al. [2011]	1024×65	-	77.3 ± 0.6	83.1 ± 0.7
Feature coding with restricted Boltzmann machine					
Sparse RBM	Sohn et al. [2011b]	4096	68.6	74.9	-
CRBM	Sohn et al. [2011b]	4096	71.3	77.8	-
Feature coding with supervised dictionary learning					
Discriminative sparse coding	Boureau et al. [2010a]	2048	-	-	85.6 ± 0.2
LC-KSVD	Jiang et al. [2011]	1024	67.7	73.6	-
*Proposed **single-layer** feature coding (Section 5.2)*					
Unsupervised (SIFT)	Goh et al. [2012]	1024	66.8 ± 1.6	75.1 ± 1.2	84.1 ± 0.8
Fine-tuned (SIFT)	Goh et al. [2012]	1024	67.5 ± 1.2	75.7 ± 1.1	84.3 ± 0.6
Unsupervised (macro feature)	Goh et al. [2012]	1024	70.2 ± 1.9	78.0 ± 1.4	85.7 ± 0.7
Fine-tuned (macro feature)	Goh et al. [2012]	1024	71.1 ± 1.3	78.9 ± 1.1	86.0 ± 0.5
*Proposed **hierarchical** feature coding (Section 5.3)*					
Unsupervised (SIFT)	Goh et al. [2014]	2048	62.5 ± 1.4	69.9 ± 1.2	79.6 ± 0.5
Fine-tuned (SIFT)	Goh et al. [2014]	2048	69.3 ± 1.1	77.2 ± 0.8	85.2 ± 0.5
Unsupervised (macro feature)	Goh et al. [2014]	2048	65.3 ± 1.5	72.8 ± 1.1	82.5 ± 0.6
Fine-tuned (macro feature)	Goh et al. [2014]	2048	$\mathbf{72.1 \pm 1.3}$	$\mathbf{79.7 \pm 0.9}$	$\mathbf{86.4 \pm 0.6}$

Comparison with other methods. Table 5.3 presents a comparison with other methods that focus on other (non-feature-coding) aspects of image categorization. The proposed hierarchical feature coding scheme outperforms all recent pixel-based convolutional methods. The main difference is in the hybridization of deep learning and the BoW model, whereby the descriptive power of SIFT, the image modeling by the spatial pyramidal pooling and robustness of SVM classification, are all exploited. This is also particularly useful when there are few labeled training examples to learn from, so hand-crafted descriptors form good surrogates for low-level representations.

Also shown in Table 5.3, are other methods that focus on visual information processing operations after the feature coding step. Amongst these methods, two recent methods by Duchenne et al. [2011] and Feng et al. [2011] reported impressive performances on the Caltech-101 dataset, which are currently the best accuracies on the dataset. Duchenne et al. [2011] used graph matching to encode the spatial information of representations learned by decoding networks [Boureau et al., 2010a]. Feng et al. [2011] built upon LLC sparse codes [Wang et al., 2010] to perform pooling in a discriminative manner, using an ℓ_p norm aggregation strategy to pool codes in between average and max, and combined with a spatial weighting term optimized for classification. These methods [Duchenne et al., 2011; Feng et al., 2011], rely on existing feature coding schemes and address the image classification problem in a completely different direction. As a result, the methods can be easily incorporated in the BoW model and possibly boost performances when combined with the proposed hierarchical feature coding schemes.

TABLE 5.3: Comparison with non-feature-coding methods on the Caltech-101 dataset.

Method	Authors	Caltech-101	
		15 tr. img.	30 tr. img.
Proposed hierarchical feature coding (Section 5.3)			
Fine-tuned (SIFT)	Goh et al. [2014]	69.3 ± 1.1	77.2 ± 0.8
Fine-tuned (macro feature)	Goh et al. [2014]	72.1 ± 1.3	79.7 ± 0.9
Convolutional networks			
Convolutional deep belief net	Lee et al. [2009]	57.7 ± 1.5	65.4 ± 0.5
Convolutional network	Kavukcuoglu et al. [2009]	-	66.3 ± 1.5
Deconvolutional network	Zeiler et al. [2010]	58.6 ± 0.7	66.9 ± 1.1
Hierarchical sparse coding	Yu et al. [2011]	-	74.0 ± 1.5
Post-feature-coding methods			
NBNN	Boiman et al. [2008]	65.0 ± 1.1	70.4
SVM-KNN	Zhang et al. [2006]	59.1 ± 0.6	66.2 ± 0.5
NBNN kernel	Tuytelaars et al. [2011]	69.2 ± 0.9	75.2 ± 1.2
Graph-matching kernel	Duchenne et al. [2011]	75.3 ± 0.7	80.3 ± 1.2
GLP	Feng et al. [2011]	70.3	82.6

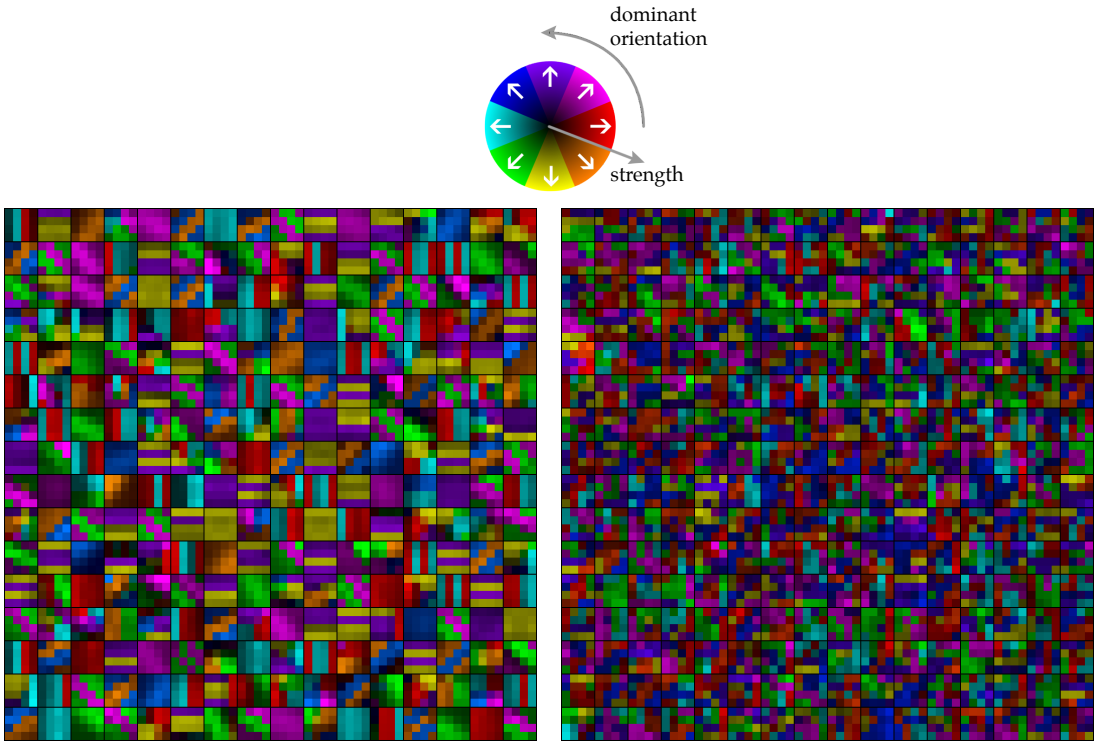
Inference speed. Due to the learned encoder of the RBM, feature coding is fast during inference. The two tier hierarchy is simply a feedforward network that performs feedforward computation twice to obtain its coding. When implemented, descriptors can be computed concurrently in batches. The unsupervised visual dictionary has the same inference time as the supervised fine-tuned version. The advantage of inference speed is especially significant when compared against decoder networks [Yang et al., 2009; Wang et al., 2010; Boureau et al., 2010a], which have to re-run the sparse coding optimization during inference. Experimentally, an inference speedup of 80 times over the ScSPM method [Yang et al., 2009] is recorded for the feature coding step.

5.5.2 Analysis: single-layer feature coding

Visualization of visual words. As described in Section 3.4.1, each visual word $\mathbf{w}_j$ can be visualized in the input feature space (Figure 3.9). In this case, the visualization is done in the space of the SIFT descriptor. After training a single-layer visual dictionary on SIFT descriptors extracted from the Caltech-101 dataset, each visual word is extracted as a filter over the 128-dimensional SIFT feature space. Each SIFT descriptor encodes quantized orientated gradients, in 8 bins, within a 4×4 grid (see Figure 2.18). The filter is partitioned into the same spatial grid and the dominant orientation is selected for each grid. The strength of the dominant orientation is also captured based on the reconstructed response of the filter. For visualization, each orientation is assigned a distinctive hue, while the strength of the local response of the partition is indicated by the intensity value. This novel visualization method can be adapted for other gradient-based local descriptors, such as the HOG descriptor [Dalal and Triggs, 2005].

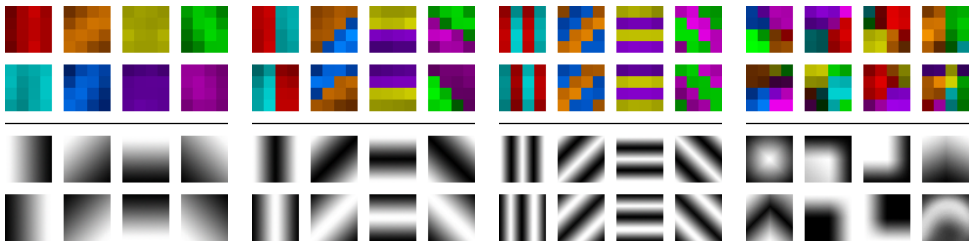
The result of the visualization is shown in Figure 5.14(a), where each square represents a visual word. It is interesting to observe that the RBM automatically discovers coherent structure in from the image gradients as compared to a normal RBM (Figure 5.14(b)). This is due to the sparse and selective regularization employed. For many visual words, opposing gradients are paired and have consistent directions. For example, red-cyan pairings tend to occur left and right of each other. A further analysis of the visualization is described in Figure 5.15.

The diversity between the visual words lead to differentiation and discrimination between features in the visual coding layer. For a single-layer visual dictionary, the visual words learned from SIFT extracted from the Caltech-101, Caltech-256 and 15-Scenes datasets are visually similar. This leads to the potential that the dictionaries learned are generic enough to be transferred between datasets in the framework of self-taught learning [Raina et al., 2007] (see Section 5.5.3).



(a) Sparse and selective RBM. (b) Standard RBM.

FIGURE 5.14: Visualization of visual words learned from SIFT descriptors. Each square represents a visual word projected back into the feature space. Each visual word is further partitioned into 4×4 neighborhoods. A partition expresses the dominant orientation as one of eight colors, while the response strength of the partition is indicated by its color intensity (see legend). By observation, the visual words trained with (a) sparsity and selectivity regularization capture more spatially coherent structure than (b) a non-regularized RBM visual dictionary.



(a) (b) (c) (d)

FIGURE 5.15: Types of gradients encoded by the visual words learned. Examples of visual words learned by the sparse and selective RBM on SIFT descriptors are shown above, and their corresponding image patches with hypothetical structure that might strongly activate the visual word are presented below. Refer to the legend of Figure 5.14. The visual words are observed to conform to image structure; most visual words encode (a) smooth gradients, (b) lines and edges, (c) textured gratings or (d) other complex features, such as corners, bends and center-surround features. The diversity between the visual words lead to differentiation and discrimination between features in the coding layer.

Visual dictionary size. Due to the sparse and selective regularization, the visual words learned encode generic image structure and tend to be very diverse and distinctive. This results in concise visual dictionaries with few redundant visual words. As a result, the method remains very competitive even as the size of the visual dictionary is reduced (see Figure 5.16). The final image signature is 32.5 times smaller than that of the best performing decoder network [Boureau et al., 2011]. The number of visual words used is also half as compared to the RBM-based method of Sohn et al. [2011b]. In both cases, the single-layer dictionary was able to outperform other methods in terms of classification performance.

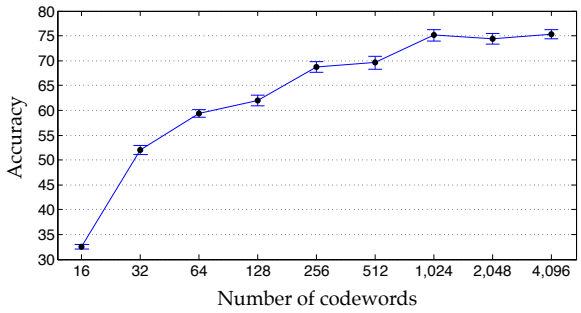


Figure 5.16: Image classification results with varying visual dictionary size. Image classification results of an unsupervised visual dictionary on Caltech-101 with 30 training examples, show that although performance degrades when the number of visual words is reduced, the results remain competitive.

Effects of sparse and selective regularization. The regularization of visual dictionaries with both sparse and selective distributions is important. When regularized with only a sparse or selective distribution, the image classification results drop significantly. On the Caltech-101 dataset using 15 training examples, using an unregularized RBM, the accuracy obtained was 51.7%. With a sparsity-only or selectivity-only regularization, the results were 45.5% and 36.8% respectively – a significant drop from the unregularized version. However, when sparsity and selectivity are jointly regularized, the classification accuracy obtained was 66.8%.

The average selectivity of the collection of visual codewords is equivalent to the sparsity of visual codes averaged across input examples. The results are analyzed using this average selectivity (or average sparsity) metric with the 15-Scenes dataset using the single-layer unsupervised visual dictionary trained on macro features. In Figure 5.17, the effects of varying levels of induced selectivity and sparsity on image classification can be observed. The classification performance suffers on both ends of the spectrum when the representation is too sparse and selective, or too densely-distributed and broady-tuned. This phenomenon is consistent with the results reported in Section 3.4.2.

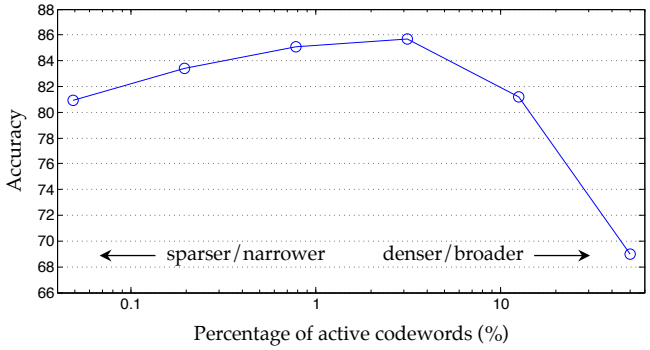


FIGURE 5.17: Effects of sparsity and selectivity on classification performance. Results on the 15-Scenes dataset with an unsupervised visual dictionary jointly regularized with sparsity and selectivity show that the performance degrades as the coding gets too sparse and narrow, or too dense and broad.

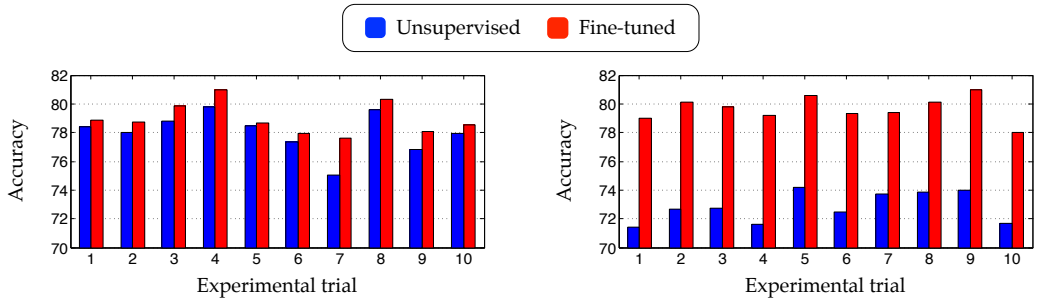
5.5.3 Analysis: hierarchical feature coding

Impact of supervision on hierarchical visual dictionaries. For each feature type – SIFT or macro feature – the hierarchical visual dictionary with supervised fine-tuning produced the best performing image classification results. However, it is not immediately apparent that either depth or supervision alone will bring significant gains to the classification performance. The empirical performance of both architectural depth and supervision will be discussed with possible explanations of the results obtained.

The analysis begins with the single-layer unsupervised model, which is already close to the state-of-the-art results, especially when encoding macro features. When the second visual dictionary is stacked, a consistent drop in the performance is observed. This may be due to the model deviating from the classification objective as layers are added – a problem that may exist even with superior generative learning on the maximum likelihood criterion. It may not be sensible to increase the depth of the model, if the resulting parameters are unable to adapt to suit the image classification task.

As shown in Table 5.2, the classification results improve when supervised fine-tuning is performed on both the single-layer and hierarchical visual dictionaries. While there is only a slight improvement when supervision is added to the single-layer coding scheme, the gains are particularly large when the visual dictionary is hierarchical, so much so that it overcomes the deficit of performance due to the depth. A conceivable reason for this is that a hierarchical architecture has the intrinsic capacity to encode more complex representations within its structure. This complexity increases class-wise modeling and separability when supervised fine-tuning is performed. This shows the importance of supervised fine-tuning, especially for a coding scheme that is several layers deep. Ultimately, it was the combination of supervised fine-tuning, coding depth and macro features that delivered the best image classification scores.

From Table 5.2, it is obvious that the gains due to supervision are statistically significant for the hierarchical model, however its benefits on the single-layer model remained uncertain. Analyses of individual trials with 30 training examples on Caltech-101 (Figure 5.18) reveal that the results of every trial improves through fine-tuning. The average improvement per trial for the single-layer and hierarchical models are $0.9 \pm 0.6\%$ and $6.8 \pm 0.6\%$ respectively. The gain is always positive for both models, but supervision is especially important to the hierarchical model. The models outperform the discriminative dictionary [Boureau et al., 2010a], which uses a more complex optimization with global labels. They also outperform the LC-KSVD model [Jiang et al., 2011], due to superior representations catalyzed by unsupervised learning.



(a) Individual trial results for the single-layer model. (b) Individual trial results for the hierarchical model.

Figure 5.18: Impact of supervised fine-tuning on feature coding. Results for 10 individual trials on Caltech-101 with 30 examples show that supervised fine-tuning consistently improves the classification results for every trial. The performance boost is more substantial for (b) the hierarchical model as compared to (a) the single-layer model.

Transfer learning between datasets. Due to the similarity between Caltech-101 and Caltech-256, I attempted to transfer the unsupervised visual dictionaries learned using Caltech-101 to classify Caltech-256 images, in the spirit of self-taught learning [Raina et al., 2007]. Meanwhile, the Caltech-256 dataset is used for supervised fine-tuning when required. The results of this evaluation is presented in Table 5.4.

With a shallow architecture, the unsupervised visual dictionary from Caltech-101 essentially performed the same as the one trained from Caltech-256. However, after stacking a second layer trained using the Caltech-101 dataset, the errors resulting from the transfer of the visual dictionary become apparent. If the first layer is trained with the Caltech-101 dataset and the second layer is trained on the Caltech-256 dataset, the results are again no different from when both layers have been trained using the Caltech-256 dataset. One theory is that the first layer models generic spatially-local dependencies. As visual codes are spatially aggregated while stacking a new layer, the representation becomes more class specific, so transferring Caltech-101 information deeper into the architecture will not be as useful as learning directly from the more complex Caltech-256 dataset itself.

TABLE 5.4: Performance on Caltech-256 for single-layer and hierarchical feature coding with transfer learning setups.

Method	Authors	Training set(s)		30 training images		60 training images	
		Layer 1	Layer 2	Unsup.	Fine-tuned	Unsup.	Fine-tuned
Proposed feature coding schemes ***without transfer learning***							
Single-layer		Caltech-256	-	41.0 ± 1.0	41.1 ± 0.8	46.1 ± 0.9	46.0 ± 0.8
Hierarchical		Caltech-256	Caltech-256	38.9 ± 0.8	41.5 ± 0.7	44.7 ± 0.8	47.2 ± 0.9
Proposed feature coding schemes ***with transfer learning***							
Single-layer		Caltech-101	-	40.8 ± 1.1	41.0 ± 1.0	45.8 ± 0.9	45.9 ± 1.0
Hierarchical		Caltech-101	Caltech-101	36.5 ± 1.0	38.3 ± 0.9	41.4 ± 1.0	44.2 ± 1.0
Hierarchical		Caltech-101	Caltech-256	39.6 ± 0.9	41.7 ± 0.9	44.0 ± 1.1	47.0 ± 1.0
Other competitive methods							
ScSPM	Yang et al. [2009]	Caltech-256		34.0		40.1	
Graph-match kernel	Duchenne et al. [2011]	Caltech-256		38.1 ± 0.6		-	
LLC	Wang et al. [2010]	Caltech-256		41.2		47.7	
CRBM	Sohn et al. [2011b]	Caltech-256		42.1		47.9	
GLP	Feng et al. [2011]	Caltech-256		43.2		-	

5.6 Summary and discussion

Two aspects of regularizing RBM-based deep networks were proposed in Chapter 3 and Chapter 4 for unsupervised greedy learning and supervised deep learning respectively. Building upon the experiences gained from modeling image patches and tackling simpler image classification problems, such as handwritten digit recognition, the deep learning strategies were employed for classifying larger images of object classes and scenes categories.

The bag-of-words (BoW) model was adopted for its visual modeling capabilities, particularly by using robust local image descriptors and spatial pyramidal pooling that take into account the spatial layout in an image. The support vector machine was also used for the final image classification. The proposed deep learning was integrated with the BoW model by exploiting their adaptability and representation power to learn hierarchical visual dictionaries for performing the feature coding of local descriptors. The spatial aggregation of lower-level features was also performed to capture spatial correlations and enhance the representations.

The resulting visual dictionaries produced leading performances on both the Caltech-101 and 15-Scenes datasets, as compared with other feature-coding-based methods. Inference speeds were also faster than the more commonly used decoder network approaches. Both the unsupervised regularization using selectivity and sparsity, as well as the use of supervised fine-tuning were shown to be crucial in enhancing the classification performances. The visual dictionaries learned were qualitatively and quantitatively determined to contain diverse and non-redundant visual words with spatially coherent image structure. The low-level visual dictionary also exhibited generality in encoding and was applied in a transfer learning framework to produce competitive results on the Caltech-256 dataset.

As an extension from this hierarchical architecture, the pooling step was also optimized and will be presented in the next chapter.

6 Discriminative Pooling

Chapter abstract — In the bag-of-words model, local visual codes have to be transformed through a pooling step to form the image signature for classification. This chapter introduces a novel generalization and parameterization of the pooling operation. The pooling parameters can be optimized via discriminative gradient descent to learn how much to pool and where to pool from. As a result, when combined with the learned hierarchical visual dictionaries (Chapter 5), the chain of operations from feature coding to pooling and classification are all optimized, one module at a time.

6.1 Introduction

Representations produced by the feature coding step of the bag-of-words (BoW) model encode an image at various spatial positions into visual codes (see Chapter 5). However, multi-dimensional data classification systems, such as neural networks (Section 2.2.1) and support vector machines [Vapnik, 1995] (Section 2.4.5), require a single fixed-sized image vector to construct the discriminative classification model that associates the image to its label. This necessitates an intermediate step between feature coding and classification, whereby the local visual codes are transformed into a single image vector or signature that describes the entire image. This is known as the pooling step (Figure 6.1).

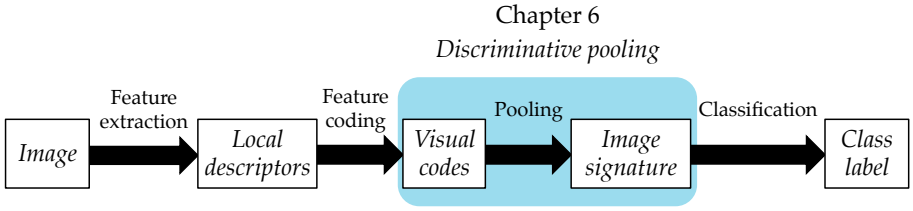


Figure 6.1: The pooling step within the bag-of-words pipeline.

Current methods typically approach this pooling step using fixed spatial partitions, whereby predefined spatial pyramids [Lazebnik et al., 2006] are typically used to define spatial partitions. The pooling scheme that summarizes the coding of the partitions into a scalar value is also set as either the max or average pooling. This chapter proposes a method to optimize this step using a discriminative criteria. While visual dictionaries were locally fine-tuned previously (Chapter 5), the objective of the pooling optimization is to bring the representations to the image-level so that the learning can be done image-wide.

6.2 Discriminative pooling optimization

To optimize the pooling step, the existing pooling method is studied and generalized. From this, a parameterization can be performed to allow for the tuning of the operation. This tuning can then be automatically optimized by introducing a discriminative criteria and performing gradient descent using the error backpropagation algorithm.

6.2.1 Generalized pooling scheme

The pooling scheme constructs the image signature by executing a sequence of two steps, namely, 1) spatial partitioning and 2) pooling (see Figure 6.2). The coded image $\mathbf{Z} \in \mathbb{R}^{J \times N}$ is coded by J visual words at N local positions. It is dissected into S spatial

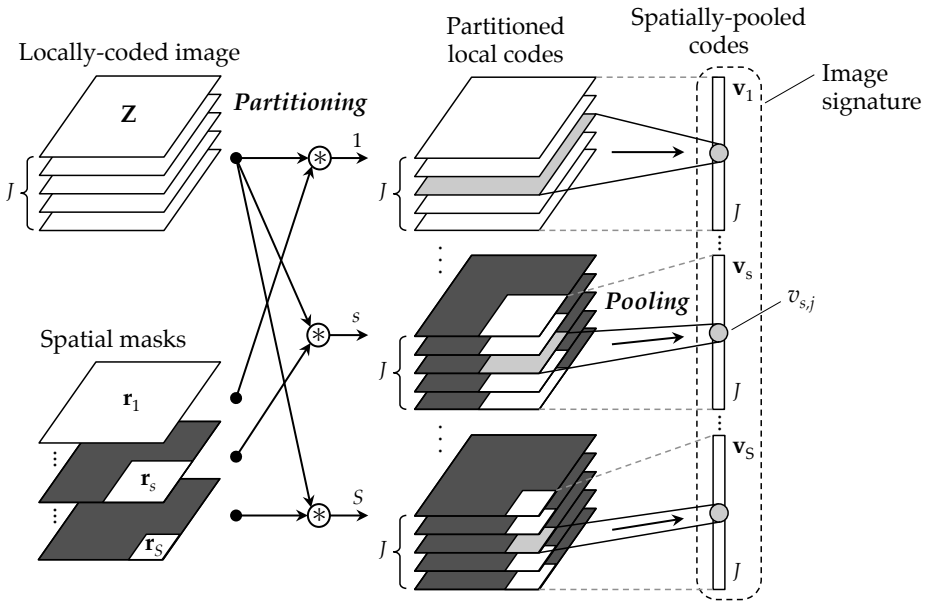


Figure 6.2: Computing the image signature by partitioning and pooling. In the partitioning step, a set of spatial masks is used to select the areas of representation for each visual coding dimension. The pooling step summarizes the visual codes within a partition into a scalar value. These scalar values are concatenated to form the image signature.

partitions. A spatial mask $\mathbf{r}_s \in \mathbb{R}^N$ is applied for each partition, where the positions n inside the partition are selected with ones and those outside are masked out with zeros. For spatial pyramids [Lazebnik et al., 2006], partitions are determined by fixing multi-scale grids over the image. The pooling operation is performed by summarizing each dimension j of the visual code across the spatial partition s. The result of pooling is a histogram of visual code occurrence $\mathbf{v}_s$ over the spatial partition defined by $\mathbf{r}_s$. Finally, the histograms are concatenated to form the final image signature, with a total of JS elements.

Traditionally, either an average or max pooling is used to compress the codes within a partition. The pooling step can be generalized by defining the pooling operation as an aggregation of visual codes z_{jn} in the image:

$$v_{s,j} = \sum_{n=1}^{N} z_{jn}\theta_{s,jn}, \tag{6.1}$$

where $\theta_{s,jn}$ is the pooling weight of z_{jn} for the spatial partition s. From this formulation, performing pooling is equivalent to computing the weighted sum of visual codes drawn across a spatial partition.

For the average pooling within a partition, the pooling weights are equal:

$$\theta_{s,jn} = \frac{r_{s,n}}{\sum_{\hat{n}=1}^{N} r_{s,\hat{n}}}. \tag{6.2}$$

If max pooling is performed, the visual code with the highest value within the partition is selected:

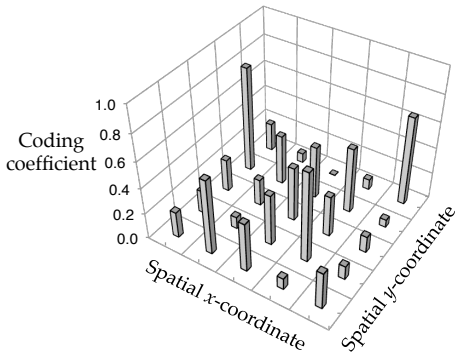
$$\theta_{s,jn} = \begin{cases} 1 & \text{if } r_{s,n}z_{jn} = \max\left\{r_{s,\hat{n}}z_{j\hat{n}}\right\}_{\hat{n}=1}^{N} \\ 0 & \text{otherwise.} \end{cases} \tag{6.3}$$

6.2.2 Parameterized pooling

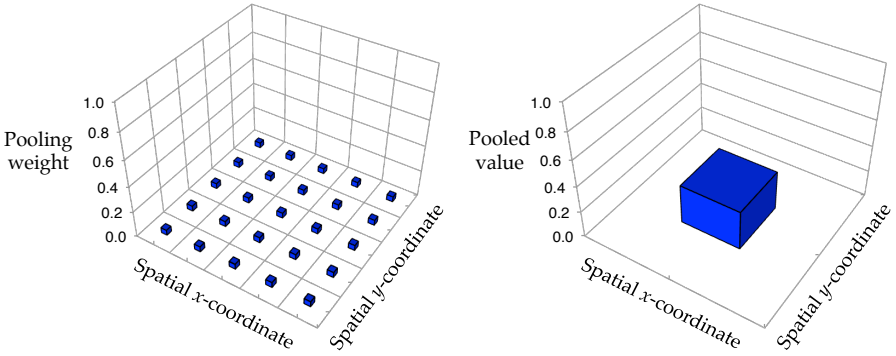
Various elements within the two steps of the pooling operation can be parameterized and eventually learned. The pooling weights can be defined by the softmax function, also known as the Boltzmann distribution:

$$\theta_{s,jn} = \frac{r_{s,n}\exp\left(\beta_{s,j}z_{jn}\right)}{\sum_{\hat{n}=1}^{N} r_{s,\hat{n}}\exp\left(\beta_{s,j}z_{j\hat{n}}\right)}, \tag{6.4}$$

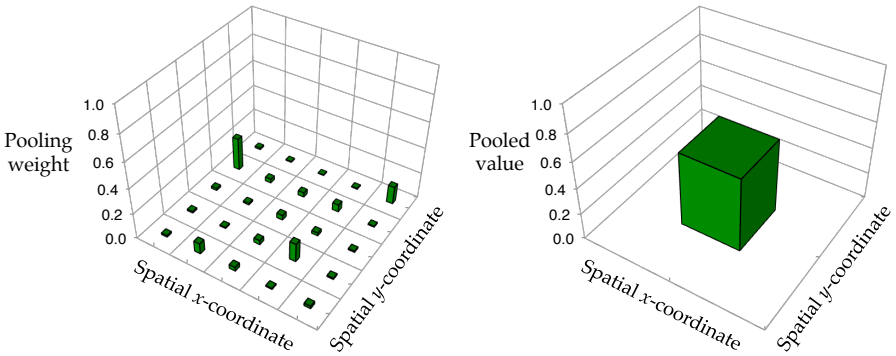
where $r_{s,n} \in [0,1]$ is the degeneracy parameter that constrains the spatial partition s, while $\beta_{s,j}$ is the inverse temperature parameter, controlling the softness of the pooling weights for the set of visual codes $\{z_{jn} : n \in [1,N]\}$ for partition s. If $\beta_{s,j} = 0$, all code selections are equally weighted, which produces the average pooling (Equation 6.2).



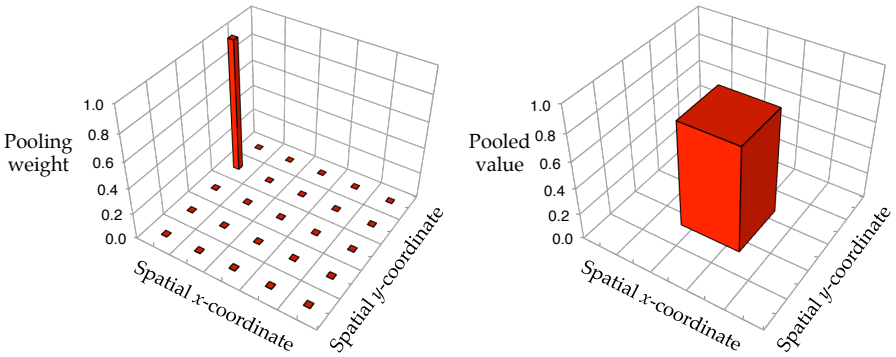
(a) Coding coefficients at various spatial positions in a partition s for a visual code j.



(b) At $\beta_{s,j} = 0$, pooling weights are equally assigned. The pooled value is the average coding value.



(c) At $\beta_{s,j} = 5$, pooling weights are differently assigned. The pooled value is between the average and max.



(d) At $\beta_{s,j} = 500$, the maximum coding in the partition is assign a weight of 1. The result is max pooling.

Figure 6.3: Pooling weights and pooled value for different parameters. The pooling step assigns pooling weights to the coding at various spatial positions based on a parameter $\beta_{s,j}$. (b) If $\beta_{s,j}$ is 0, average pooling is performed. (d) When $\beta_{s,j}$ is sufficiently large, max pooling is approximated.

The max pooling (Equation 6.3) is given by $\beta_{s,j} \to \infty$, which greedily selects the most intense coding. Figure 6.3 shows examples of pooling using different values of $\beta_{s,j}$.

The spatial mask $r_{s,n}$ can be split into two channels: a binary selector $b_{s,n} \in \{0, 1\}$ and a soft spatial mask $\alpha_{s,n} \in]0, 1]$, with

$$r_{s,n} = \alpha_{s,n} b_{s,n}. \tag{6.5}$$

The binary mask makes a hard selection of the partition by assigning ones to positions within the partition, while zeros are set at all other feature positions. The soft mask forms an alpha channel over the hard mask and can be seen as a saliency map for the spatial partition denoted by the binary mask. Each local code is weighted by $\alpha_{s,n}$ may be weighted differently for different spatial partitions. This weighting *within* a spatial partition differs from the weighting *between* partitions done by Sharma et al. [2012]. Partition weighting is not necessary in this case because the SVM classifier will learn the weights for each partition in the subsequent step of the BoW model.

Combining the pooling weights and modified spatial masks with Equation 6.1, the resulting complete parameterized pooling function can be defined as

$$v_{s,j} = \frac{\sum_{n=1}^{N} z_{jn} \alpha_{s,n} b_{s,n} \exp\left(\beta_{s,j} z_{jn}\right)}{\sum_{n=1}^{N} \alpha_{s,n} b_{s,n} \exp\left(\beta_{s,j} z_{jn}\right)}. \tag{6.6}$$

6.2.3 Discriminative pooling optimization

From the pooling parameterization, we can see that the parameters represent two different aspects of pooling. While $\beta_{s,j}$ determines what to pool, $\alpha_{s,n}$ directs where to pool from. The binary mask $b_{s,n}$ acts as a fixed region selector and is not optimized. The next objective is to optimize these parameters to learn the pooling within each partition. The softmax pooling function (Equation 6.6) is differentiable, which enables the learning of parameters by gradient descent.

A set of classification parameters $\mathbf{W}_3$ is added to link the pooled image signature to the class labels. The image can be locally encoded, pooled and has its class label predicted. This prediction can be compared with the ground truth class label through a discriminative loss function $\mathcal{L}_{disc}$, which provides the learning signal. The updates for the pooling parameters can be computed by chaining the gradients $\frac{\partial \mathcal{L}_{disc}}{\partial v_{s,j}} \frac{\partial v_{s,j}}{\partial \beta_{s,j}}$ and $\frac{\partial \mathcal{L}_{disc}}{\partial v_{s,j}} \frac{\partial v_{s,j}}{\partial \alpha_{s,n}}$ respectively, which backpropagates the learning signal. Figure 6.4 shows the backpropagated discriminative learning signals extended from the result of the hierarchical feature coding scheme described in Section 5.3.

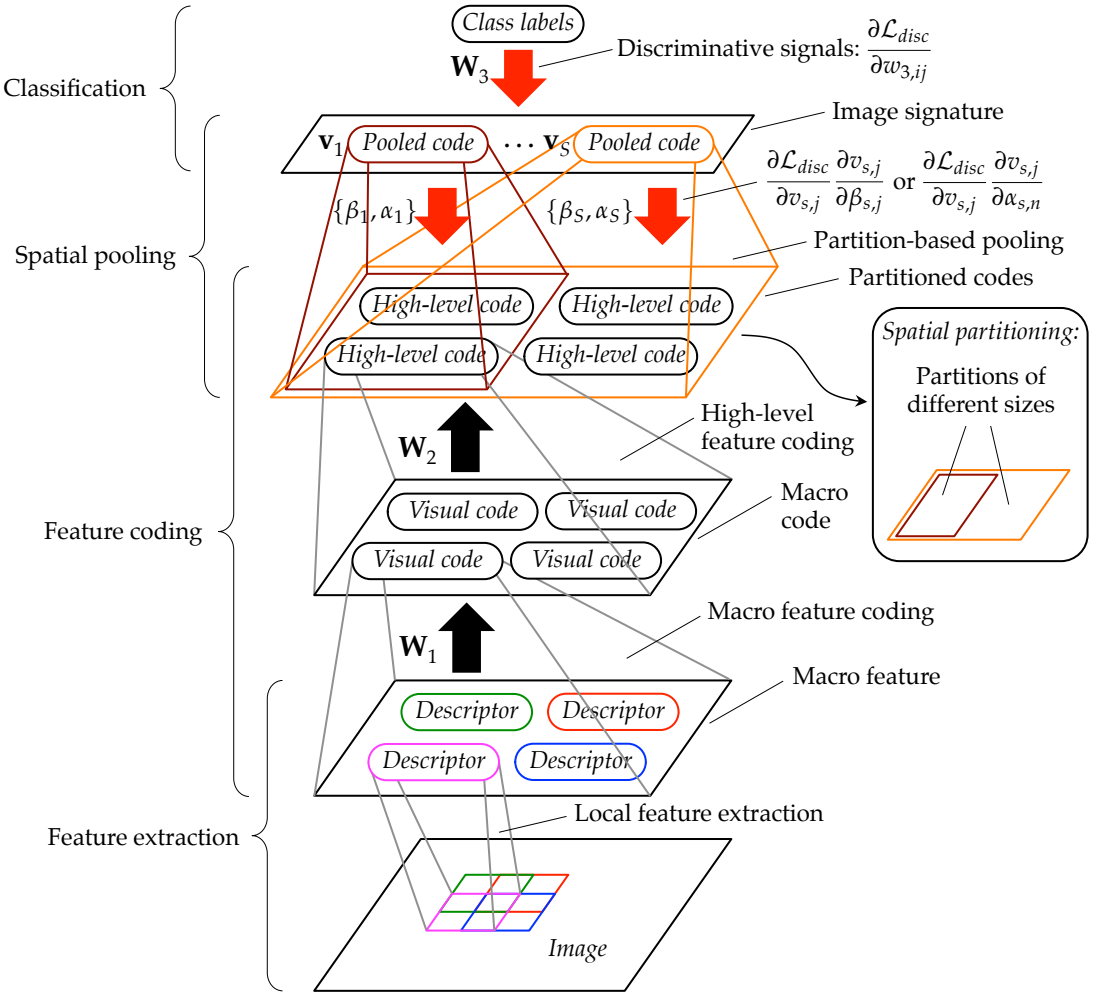


Figure 6.4: Optimizing the pooling parameters with backpropagation. Pooling is performed over different subsets of visual codes defined by the spatial partitions. Image classification errors are used to provide update the pooling parameters. Here, the feature coding is provided by a hierarchical feature coding scheme (Section 5.3).

Optimizing the code selection parameter. The inverse temperature parameter $\beta_{s,j}$ controls the softness of the coding selection. It is, therefore, coherent that the partial derivative of $v_{s,j}$ with respect to $\beta_{s,j}$ is derived to be:

$$\frac{\partial v_{s,j}}{\partial \beta_{s,j}} = \left(\sum_{n=1}^{N} z_{jn}^2 \theta_{s,jn} \right) - \left(\sum_{n=1}^{N} z_{jn} \theta_{s,jn} \right)^2 . \tag{6.7}$$

Please refer to Appendix A.4 for the derivation of this gradient.

Optimizing the code masking parameter. The binary masks are fixed and not optimized. Meanwhile, the soft mask may be modified to learn a soft spatial selection within the window defined by the binary mask. The gradient of $v_{s,j}$ with respect to the soft mask $\alpha_{s,n}$ for the local code z_{jn} is determined by its pooling weight, its deviation

from the pooled value and its current soft mask value:

$$\frac{\partial v_{s,j}}{\partial \alpha_{s,n}} = \frac{1}{\alpha_{s,n}}(z_{jn} - v_{s,j})\theta_{s,jn}. \tag{6.8}$$

The derivation of the gradient is detailed in Appendix A.5.

Optimizing the global masking model. For each image instance t, the soft mask $\alpha_{s,n}$ is assigned to each visual code at each local position n. However, images t have different sizes, so the number of visual codes $N_{s,t}$ for a spatial partition is different for each image instance. So far, the optimization is only performed to the local coding, so a spatial normalization needs to be done across the images. Resizing the images in the dataset to the same size will distort the aspect ratio of the images and may affect the resulting local gradient coding of the images. So, the spatial resizing is performed on the parameter instead. This will form a dataset-wide soft mask $\tilde{\alpha}_{s,n}$, which can be applied for images of all sizes. When performing pooling for an image, the soft mask value $\alpha_{s,n}$ at each spatial position is computed through a bilinear interpolation $\alpha_{s,n} = f_{bilinear}(\tilde{\alpha}_{s,n})$ from the global soft masks. When updating the global soft mask, the gradient with respect to $\tilde{\alpha}_{s,n}$ factors in the specific positional weights previously used in the interpolation.

6.2.4 Relationship between code selectivity and pooling

The coding and pooling steps both operated in the same lifetime domain (across instances) and hence have a close relationship, which can be analyzed. Let us assume the coding within a single spatial partition $\mathbf{z}_{jn} : b_{s,n} = 1$ to take on the power-law distribution as described in Section 3.3.2. The average magnitude of the coding coefficients can be controlled by tuning the μ parameter (Equation 3.14). When μ is lowered, selectivity increases. Meanwhile, the $\beta_{s,j}$ parameter that controls the selection of these codes using Equation 6.6 can also be varied. As $\beta_{s,j}$ increases, the pooling function tends towards selecting the max coding value. As a result, the pooled value for different softness of code selection can be computed for each coding average μ. This relationship can be plotted in Figure 6.5.

Importance of code selectivity. When the code is highly selective, the variation of the pooled values for different $\beta_{s,j}$ is the greatest. When the code is broadly tuned, performing average or max pooling shows minimal difference and it makes no sense to attempt to optimize the pooling parameters. This shows the importance of having a selective coding, if the pooling parameter is to be optimized.

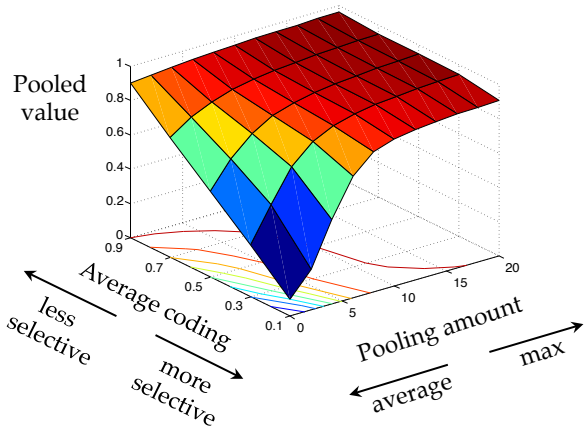


Figure 6.5: Relationship of code selectivity and pooling. High selectivity shows the most capacity for learning the pooling parameter. Max-pooling shows the least sensitivity towards the selectivity of coding.

Importance of max-pooling. If the pooling value is fixed to perform average pooling, the resulting pooled value is sensitive to the level of selectivity in the code. However, if max-pooling is used, there is little difference between the level of selectivity in the coding. As a result, max-pooling might be suitable for reducing the sensitivity towards the level of selectivity of the coding if no optimization of the pooling parameters is to be attempted.

6.3 Discriminative pooling experiments

Experiments were performed to empirically study the effects of optimizing the pooling step for image classification and analyze the optimized parameters.

6.3.1 Evaluation: image classification

Experimental datasets and setup. The experimental datasets used for this set of experiments are the 15-Scenes [Lazebnik et al., 2006] and the Caltech-10 [Fei-Fei et al., 2004] datasets as described in Section 5.4.1, with the same evaluation metric of average class-wise accuracy used (see Section 5.4.2). Two visual dictionary setups were used for the experiments. The first is the best performing hierarchical visual dictionary proposed in Chapter 5 (2048 dimensions), while the second is the sparse coding spatial pyramid matching model of Yang et al. [2009] (1024 dimensions).

After performing the feature coding step, the pooling parameters $\tilde{\alpha}_{s,n}$ and $\beta_{s,j}$ were separately optimized. Spatial pyramid with three scales: $\{[1 \times 1], [2 \times 2], [4 \times 4]\}$ was used to partition the images. The dataset-wide masking parameter $\tilde{\alpha}_{s,n}$ was sized at 4×4 global parameters for each spatial partition. The initialization for $\tilde{\alpha}_{s,n}$ was 1,

while the initialization of $\beta_{s,j}$ was set at the point of maximum gradient, which is at $\beta_{s,j} = 0$ at average pooling.

Because this optimization relies on the class labels, a training set of images per class was randomly sampled from the datasets (30 for the Caltech-101 dataset and 100 for the 15-Scenes dataset). Where supervised learning was used to fine-tune the visual dictionary, the training set used to discriminatively optimize the pooling was the same. This training set was also used to learn the classification model in the subsequent step. The training data was grouped into mini-batches, with each mini-batch having one randomized example from each class. The parameters were updated after every mini-batch.

A linear classifier with a softmax activated final layer was used to associate the image signature to the output vector $\mathbf{y} \in \mathbb{R}^C$. This classifier was concurrently trained, while optimizing the pooling parameters. The number of weight parameters for the classifier is $SJ \times C$, which is $4,386,816$ parameters when modeling the 102 classes of the Caltech-101 dataset using a three-scale spatial pyramid (21 spatial masks) to partition the images and the 2048-dimensional visual code produced by the hierarchical feature coding scheme (Section 5.3).

Evaluation results. The results of optimization are reported in Table 6.1. When optimizing the parameters $\beta_{s,j}$ that define the softness of the pooling function, there was generally a very slight, but perhaps negligible, improvement in image classification accuracy over the max-pooling method. Interestingly, yet unfortunately, the image classification results suffered significantly when attempting to optimize the soft spatial partitioning masks $\tilde{\alpha}_{s,n}$.

Table 6.1: Image classification results with optimized spatial pooling.

Experimental setup	Hierarchical feature coding (Section 5.3)	ScSPM [Yang et al., 2009]
15-Scenes (100 training images)		
Non-optimized max-pooling	86.4 ± 0.6	80.3 ± 0.9
After optimizing $\beta_{s,j}$	86.3 ± 0.6	80.6 ± 0.8
After optimizing $\tilde{\alpha}_{s,n}$	78.9 ± 0.9	74.5 ± 1.0
Caltech-101 (30 training images)		
Non-optimized max-pooling	79.7 ± 0.9	73.2 ± 0.5
After optimizing $\beta_{s,j}$	79.8 ± 0.9	73.6 ± 0.6
After optimizing $\tilde{\alpha}_{s,n}$	75.4 ± 1.1	69.8 ± 0.8

One suggestion on the lack of improvement in the image classification results is the possibility of learning an over-fitted model at the classification layer. The classifier is

huge, with parameters numbering in the millions. However, there are only 3060 and 1500 training examples for the Caltech-101 and 15-Scenes datasets, respectively. This is aggravated with the error backpropagation algorithm, since little error will then by propagated backward to tune the pooling parameters. The learning rate had to be set very high – in the order of thousands – to modify the parameters. In addition, the resulting values of $\tilde{\alpha}_{s,n}$ were found to be extremely low, which may also lead to bad generalization.

6.3.2 Analysis: how much to pool?

The 15-Scenes dataset was used to study the softness of the pooling function, defined by the $\beta_{s,j}$ parameter. After training, the values of the learned parameters were analyzed. Figure 6.6 shows the mean value of $\beta_{s,j}$, averaged across all the J visual codes for each spatial partition. An interesting correlation to the spatial scale emerged after the optimization. The learned parameters suggest that for a larger scale (1×1), a pooling that is close to max-pooling may be suitable. On the other hand, when the spatial partitions are small, the model suggests a pooling that is closer to the average pooling. This supports the observation by Boureau et al. [2010a] that max-pooling has a larger variance in the estimate than average pooling and hence average pooling works better for smaller image regions.

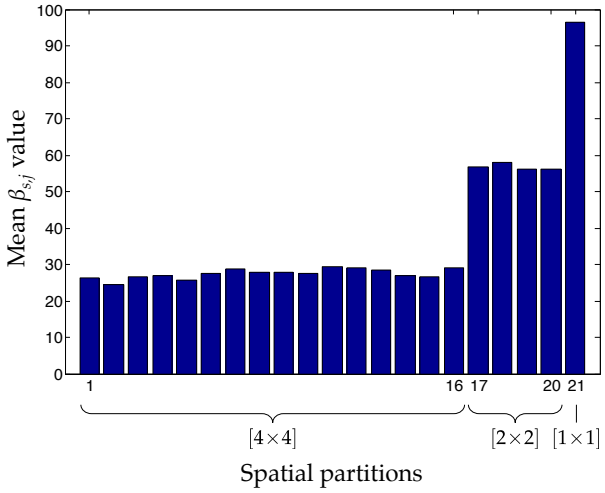
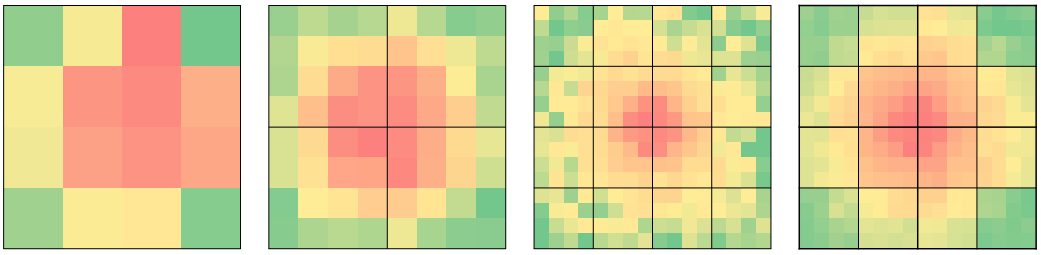


Figure 6.6: Analysis of pooling softness parameter. The softness parameter is learned to be related to the scale at which the pooling is performed in. The larger the scale, the model suggests a pooling closer to the max.

6.3.3 Analysis: where to pool from?

The Caltech-101 dataset was used to analyze the learned spatial soft mask parameters. The resulting visualization for the soft mask values are shown in Figure 6.7.

As observed, there is a strong bias towards the center of the image for this dataset. This is aligned with the claims made by Ponce et al. [2006]. What is learned is general statistic of the average positions of discriminative features in the dataset, images of objects from different instances and classes may have their own ideal parameters. This suggests that optimizing the parameter may not boost classification scores (Table 6.1). It is also observed that the soft mask values were extremely low and this may be the cause of undesirable results, which could potentially be improved by introducing normalization constraints.



(a) Scale: 1×1. (b) Scale: 2×2. (c) Scale: 4×4. (d) Combined.

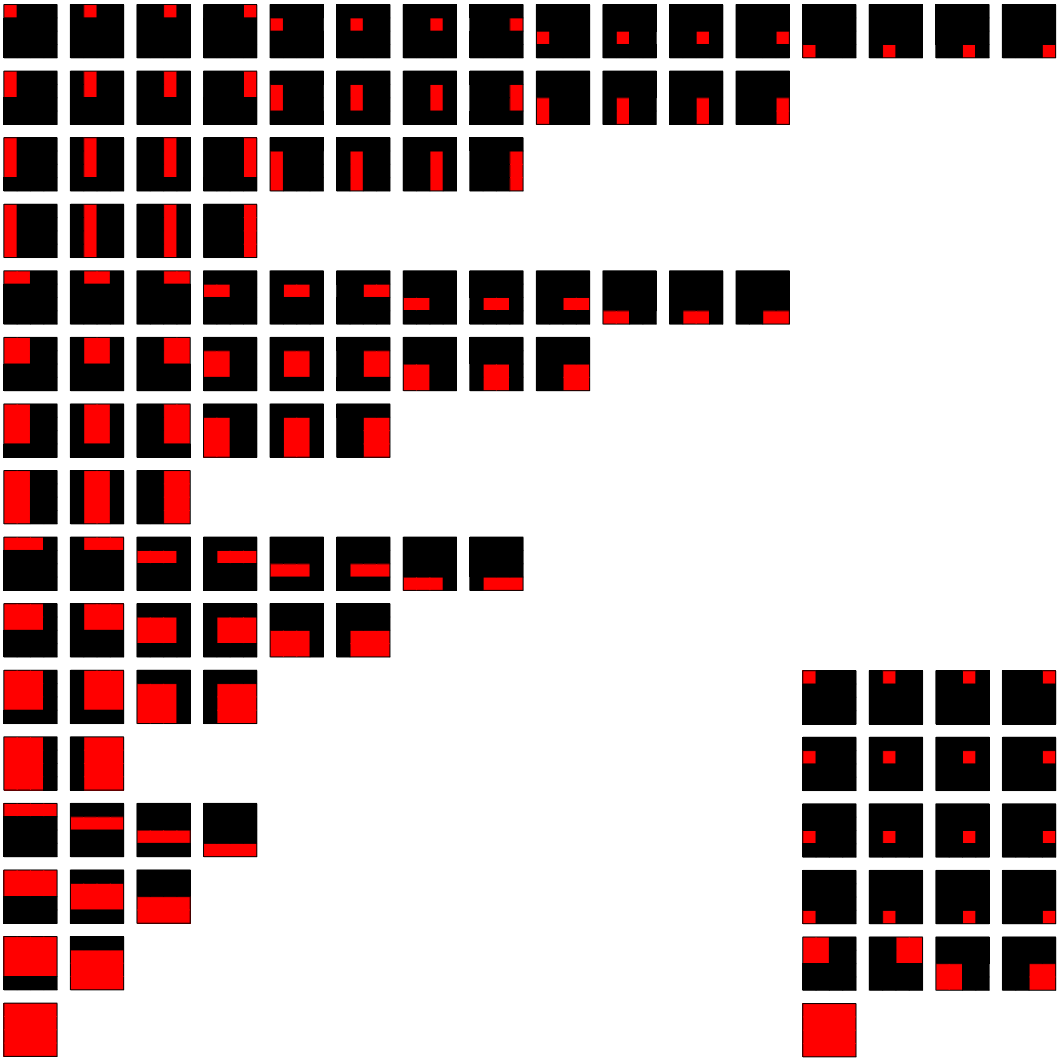
FIGURE 6.7: Visualization of learned spatial soft masks. For the Caltech-101 dataset, the global soft masks $\tilde{\alpha}_s$, each in the form of a 4×4 grid is visualized for the partitions of a spatial pyramid of the three scales: (a) $[1 \times 1]$, (b) $[2 \times 2]$ and (c) $[4 \times 4]$. (d) The resulting summation across the scales shows a bias towards the center of the image.

6.4 Potential methodological extensions

The generality of the differentiable pooling function can be exploited and extended in several directions. The following three possible extensions are proposed:

1. Learning from an exhaustive set of partitions,
2. Global discriminative fine-tuning of visual dictionaries, and
3. Optimized vectorial pooling scheme.

Learning from an exhaustive set of partitions. Instead of defining a small subset of spatial partitions to pool over, such as the spatial pyramidal scheme (Figure 6.8(b)), one can use a more generic method to generate the spatial masks, such as using an exhaustive set of partitions [Jia et al., 2012] (Figure 6.8(a)). An exhaustive partitioning scheme generates all possible rectangular partitions of various sizes within a given grid. Optimization algorithms can then be used to distinguish the important partitions. In this way, the model is less biased towards the method of partitioning. It is, however, crucial to manage the possibility of over-fitting the data because of the significant increase in number of parameters.



(a) Exhaustive partitions. (b) Pyramidal partitions.

Figure 6.8: Spatial partitioning schemes. (a) A exhaustive (or overcomplete) set of 100 partition configurations with 16 different partition types (rows) in various spatial positions (columns) within a 4×4 grid. (b) The commonly used three-level spatial pyramidal partitioning produces a subset of the exhaustive set of partitions.

Global discriminative fine-tuning of visual dictionaries. As opposed to local fine-tuning previously performed in Section 5.3.3, the visual dictionaries learned can be fine-tuned based on the exact optimization problem of classifying the image globally. The gradient with respect to to the coding parameters $\mathbf{W}$ can be derived by backpropagating the class prediction errors through the classifier and the pooled codes to the visual dictionary $\frac{\partial \mathcal{L}_{disc}}{\partial v_{s,j}} \frac{\partial v_{s,j}}{\partial z_{jn}} \frac{\partial z_{jn}}{\partial w_{ij}}$. However, it should be noted that since the error gradient is backpropagated through many layers, the same challenges of learning a deep architecture using this algorithm will apply (see Section 2.2.1 for a discussion).

Optimizing vectorial pooling. Instead of producing a single scalar value per code, one extension is to partition the space of the coding value into a number Q of partitions

and selectively pool coding values that fall within each partition [Avila et al., 2013]. The partition mask $c_{j,q}$ can be defined as follows:

$$c_{j,q} = \begin{cases} 1 & \text{if } z_q^{\min} < z_{jn} \leq z_q^{\max} \\ 0 & \text{otherwise,} \end{cases} \tag{6.9}$$

where $z_q^{\min}$ and $z_q^{\max}$ are hyperparameters dictating the lower and upper boundaries of the bin. The resulting parameterized pooling function can then be modified to incorporate this coding space partitioning:

$$v_{s,j,q} = \frac{\sum_{n=1}^{N} z_{jn} \alpha_{s,n} b_{s,n} c_{j,q} \exp\left(\beta_{s,j} z_{jn}\right)}{\sum_{n=1}^{N} \alpha_{s,n} b_{s,n} c_{j,q} \exp\left(\beta_{s,j} z_{jn}\right)}. \tag{6.10}$$

The overall representation will increase by a factor of Q. However, our preliminary studies have not yielded significant improvement over max pooling even with the increased dimensionality of the representation. It will be interesting to perform a further study on the possibility of optimizing this type of pooling to enhance image classification performances.

6.5 Summary and discussion

This chapter proposed a method to parameterize and optimize the pooling step within the bag-of-words model. Optimization is performed via gradient descent using back-propagated errors. Pooling is performed in two steps: spatial partitioning and pooling, resulting in two types of pooling parameters being learned. The first denotes the softness of the selection of the coding within a partition and a pooling amount between the commonly used average- and max-pooling strategies can be discovered. The second parameter limits the amount of pooling for different spatial positions within a spatial partition. However, the results on image classification do not do justice to the neatness of the parameterized pooling function. It is suspected that this is due to the lack of training examples in the dataset, relative to the number of parameters in the classifier, leading to over-fitting and poor generalization. There is good potential for further explorations in this direction.

7 Conclusions

Today, the advancements in the learning of deep architectures and the modeling of visual representations have brought about the fusion of the two research topics in artificial intelligence. On one hand, the bag-of-words model provides a reliable pipeline to perform image classification through a series of data transformation modules. On the other hand, deep learning techniques provide a means to adapt a complex model with strong representational power. This potential and opportunity for hybridization was one of the main motivations leading to the start of this thesis.

7.1 Learning deep visual representations: a synthesis of ideas

The problem of deep learning was studied and approached at both the micro and macro levels. At the micro level, the learning problem revolves around the automatic discovery of interesting latent representations through unsupervised learning and regularization. The material in Chapter 3 proposed a new generic regularization algorithm to complement the maximum-likelihood-based learning of restricted Boltzann machines. The generality of the regularization method when learning latent representations enables the incorporation of various representational properties, such as sparsity, selectivity and topographic organization.

At the macro level, the deep architectures were optimized as a whole, by integrating unsupervised feature learning with supervised learning, as discussed in Chapter 4. The notion of a smooth transition from a fully-unsupervised learning phase to a strongly-supervised optimization was introduced. Two methods to support this notion were designed to integrate bottom-up and top-down information. The first method incorporates supervision into the deep belief network by regularizing the layers with top-down sampled signals. The second method uses a collection of loss functions to encourage output prediction, input reconstruction to train a stack of encoder-decoder networks. The two supervised learning concepts were further embedded into

a gradually-transisting three-phase learning strategy that first performs layer-wise unsupervised learning, followed by top-down regularization and finally discriminative fine-tuning.

Using the new deep learning algorithms developed, the problem of image classification was tackled in the framework of the bag-of-words model. Chapter 5 described a hierarchical visual dictionary learned from local descriptors extracted from the images. The dictionary enables the encoding of the descriptors into more descriptive spatially-aggregated representations. The three-phase learning strategy was adopted, with regularization applied to the layer-wise unsupervised learning to obtain sparse and selective representations, and to the network as a whole to fine-tune the model using supervision. The results obtained were competitive against other dictionary learning methods, with the visual dictionary being concise and non-redundant. Inference can also be performed very quickly. Chapter 6 introduced a further attempt to extend the model by discriminatively optimizing the pooling step of the bag-of-words model so that the entire chain of operations from feature coding, to pooling and finally to classification are all optimized.

7.2 Future work

Let us return to the roadmap of this thesis (Figure 1.2) presented in Chapter 1. The following extensions to this body of work are proposed (Figure 7.1):

- Extensions on discriminative pooling,
- Fully-optimized bag-of-words model, and
- Globally-optimized convolutional deep networks.

Globally-optimized convolutional deep networks. With the recent successes of deep convolutional networks, another potential research direction is to apply the proposed deep learning methods to model images using these networks. Both bottom-up and top-down information can be exploited to enhance the learning process in terms of the quality of representations and speed of training. From my experiments, it seems that image classification performances are sensitive to the spatial sub-sampling parameter, where smaller sub-sampling step size typically produces the higher classification scores. This makes convolutional approaches to be very attractive, because it is the densest spatial sampling possible. It will also be interesting to see how pooling optimizations can be incorporated into the convolutional network.

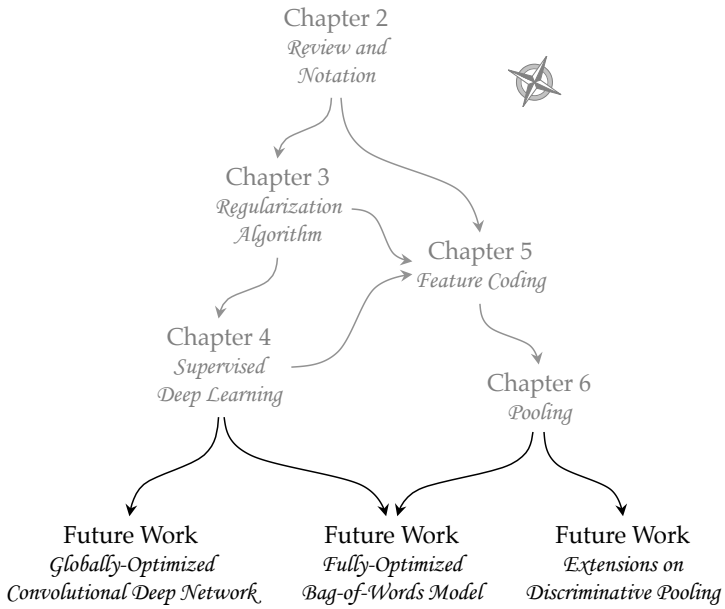


FIGURE 7.1: Roadmap to future work.

Fully-optimized bag-of-words model. This thesis introduced the notion of optimizing multiple successive steps of the bag-of-words model, from feature coding, to pooling and classification. This can also be seen as a form of learning a deep architecture that is specific for image classification. From the current model, a potential future direction may be to parameterize and optimize the feature extraction step. Already, there are similarities drawn between some deep convolutional networks and local feature extraction (see Section 2.4.1). An even more ambitious approach is to go beyond learning each module independently, but also optimizing the interactions between the modules to enhance the quality of representations (as suggested in Section 6.4).

Extensions on discriminative pooling. The discriminative pooling approach described in Chapter 6 is still very much in its infancy. There are various extensions that can be made to enhance the algorithms for this step, such as those described in Section 6.4. One bottleneck of the optimization problem is the potential over-fitting of the model's parameters. Overcoming this issue is one of the most pressing work to be done. A possible direction will be to use latent support vector machine optimizations to generate instance-based tuning of the parameters.

As Mme. Marie Curie once wrote, "*On ne fait jamais attention 'a ce qui a été fait; on ne voit que ce qui reste à faire (One never notices what has been done; one can only see what remains to be done)*". Until the problems of machine learning and computer vision are comprehensively solved, there will continue to be countless future theses romancing these two dynamic fields of research. May this marriage be an everlasting one.

7.3 List of publications

The material reported in this thesis was the subject of the following publications:

Journal Articles

- Goh, H., Thome, N., Cord, M., and Lim, J.-H. (2014). Learning deep hierarchical visual feature coding. *IEEE Transactions on Neural Networks and Learning Systems*. [Goh et al., 2014]

Conference Papers

- Goh, H., Thome, N., Cord, M., and Lim, J.-H. (2013). Top-down regularization of deep belief networks. *Advances in Neural Information Processing Systems (NIPS)*. [Goh et al., 2013]
- Goh, H., Thome, N., Cord, M., and Lim, J.-H. (2012). Unsupervised and supervised visual codes with restricted Boltzmann machines. In *European Conference on Computer Vision (ECCV)*. [Goh et al., 2012]
- Goh, H., Kuśmierz, Ł., Lim, J.-H., Thome, N., and Cord, M. (2011). Learning invariant color features with sparse topographic restricted Boltzmann machines. In *International Conference on Image Processing (ICIP)*. [Goh et al., 2011]
- Goh, H., Thome, N. and Cord, M. (2010). Biasing restricted Boltzmann machines to manipulate latent selectivity and sparsity. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*. [Goh et al., 2010a]

Others

- Goh, H., Thome, N., Cord, M., and Lim, J.-H. (2010). Neuroscience-informed sparsity and selectivity in restricted Boltzmann machines. Poster presented at *Decade of Mind VI Conference (DOM)*. [Goh et al., 2010b]

A Derivation of Gradients

Summary — In this thesis, the predominant method for learning and optimization representations is the gradient descent method. In this appendix, I detail the derivations of the partial derivatives for some of the optimizations discussed in the earlier chapters.

A.1 Point- and instance-wise regularization

The update rules for unsupervised RBM learning (Equation 3.10) consist of the combination of maximum likelihood approximation and point-wise and instant-wise regularization. We begin with Equations 3.1 and 3.9 in Chapter 3, where following optimization problem was posed:

$$\arg\min_{\mathbf{W}} \; - \sum_{k=1}^{|\mathcal{D}_{train}|} \log P(\mathbf{x}_k) + \lambda h(\mathbf{z}), \tag{A.1}$$

where λ is a regularization constant

$$h(\mathbf{z}) = \sum_{k=1}^{|\mathcal{D}_{train}|} \sum_{j=1}^{J} \mathcal{L}(z_{jk}, p_{jk}), \tag{A.2}$$

and $\mathcal{L}(z_{jk}, p_{jk})$ is simply the cross-entropy loss between the data-sampled activation z_{jk} and the target activation p_{jk}:

$$\begin{aligned} \mathcal{L}(z_{jk}, p_{jk}) &= -\log P(p_{jk}|z_{jk}) \\ &= -p_{jk} \log z_{jk} - (1 - p_{jk}) \log(z_{jk} - 1). \end{aligned} \tag{A.3}$$

Given a training example k, let the total input for z_{jk} be $u_{jk} = \sum_{i=0}^{I} w_{ij} x_{ik}$ (i.e. $z_{jk} = \sigma(u_{jk})$). To compute the update to w_{ij} due to the cross-entropy loss, we take the partial

derivative of $\mathcal{L}(z_{jk}, p_{jk})$ with respect to w_{ij} and apply a negative constant $-\eta \propto \lambda$ to indicate the intention to reverse the error of z_{jk}:

$$\begin{aligned}
\Delta w_{ij}(k) &= -\eta \frac{\partial \mathcal{L}(z_{jk}, p_{jk})}{\partial w_{ij}} \\
&= -\eta \frac{\partial \mathcal{L}(z_{jk}, p_{jk})}{\partial z_{jk}} \frac{\partial z_{jk}}{\partial u_{jk}} \frac{\partial u_{jk}}{\partial w_{ij}} \\
&= -\eta \left(\frac{1 - p_{jk}}{1 - z_{jk}} - \frac{p_{jk}}{z_{jk}} \right) \left(z_{jk} \right) \left(1 - z_{jk} \right) x_{ik} \\
&= -\eta \frac{(1 - p_{jk})(z_{jk})(1 - z_{jk})}{1 - z_{jk}} - \frac{p_{jk}(z_{jk})(1 - z_{jk})}{z_{jk}} x_{ik} \\
&= -\eta \, x_{ik}(z_{jk} - p_{jk})
\end{aligned} \tag{A.4}$$

Together with the original update rule of the RBM (Equation 2.36), the batch-wise parameter update with point-wise and instance-wise regularization (Equation 3.10) may be obtained:

$$\begin{aligned}
\Delta w_{ij} &= \varepsilon \left(\langle x_i z_j \rangle_{data} - \langle x_i z_j \rangle_{recon} \right) - \eta \left\langle x_{i,data} \left(z_{j,data} - p_j \right) \right\rangle \\
&= \varepsilon \left(\langle x_i z_j \rangle_{data} - \frac{\eta}{\varepsilon} \left\langle x_{i,data} \left(z_{j,data} - p_j \right) \right\rangle - \langle x_i z_j \rangle_{recon} \right) \\
&= \varepsilon \left(\left\langle x_{i,data} \left(z_{j,data} - \frac{\eta}{\varepsilon} \left(z_{j,data} - p_j \right) \right) \right\rangle - \langle x_i z_j \rangle_{recon} \right) \\
&= \varepsilon \left(\left\langle x_{i,data} \left(\left(1 - \frac{\eta}{\varepsilon} \right) z_{j,data} + \frac{\eta}{\varepsilon} p_j \right) \right\rangle - \langle x_i z_j \rangle_{recon} \right) \\
&= \varepsilon \left(\langle x_{i,data} s_j \rangle - \langle x_i z_j \rangle_{recon} \right)
\end{aligned} \tag{A.5}$$

where $s_{jk} = (1 - \phi) \, z_{jk,data} + \phi p_{jk}$ and $\phi = \frac{\eta}{\varepsilon}$ as a hyperparameter.

A.2 Squared loss penalty on activation averages

Lee et al. [2008] used a regularization term based on the squared loss as follows:

$$h(\mathbf{z}) = \sum_{j=1}^{J} \left\| \widetilde{p} - \langle z_j \rangle \right\|^2 . \tag{A.6}$$

The gradient of the regularization term is computed as:

$$\Delta w_{ij} \propto \left(\widetilde{p} - \langle z_j \rangle \right) \left\langle x_i z_j \left(z_j - 1 \right) \right\rangle . \tag{A.7}$$

According to Lee et al. [2008], only the update to the offset terms w_{0j} are required since they alone can control the level of latent activation.

$$\Delta w_{0j} \propto \left(\widetilde{p} - \langle z_j \rangle \right) \left\langle z_j \left(z_j - 1 \right) \right\rangle . \tag{A.8}$$

However, this may result in highly negative biases to induce the required degree of selectivity and highly positive weights to ensure the hidden units remain useful [Hinton, 2010]. Another drawback of this method is that as z_{jk} nears 0 or 1, the derivative of the sigmoid function approaches 0, thus degrading the learning signal. To prevent this problem, a further simplification is performed when implemented, by dropping the latter term as reported by Ekanadham [2007]:

$$\Delta w_{0j} \propto \widetilde{p} - \langle z_j \rangle . \tag{A.9}$$

A.3 Cross-entropy penalty on decaying activation averages

Nair and Hinton [2009] suggested using the cross-entropy measure between the observed and desired distributions to penalize non-selective latent variables. This discussion begins from (3.6) and (3.7). For the current training mini-batch t, this is given by

$$\begin{aligned} h(\mathbf{z}) &= \sum_{j=1}^{J} \mathcal{L}\left(q_{j,t}, \widetilde{p}\right) \\ &= \sum_{j=1}^{J} -\widetilde{p} \log q_{j,t} - (1-\widetilde{p}) \log\left(1 - q_{j,t}\right) \end{aligned} \tag{A.10}$$

where the exponentially decaying average of the mean probability is defined as [see Hinton, 2010]:

$$q_{j,t} = (1-\tau) \langle z_j \rangle_t + \tau q_{j,t-1}. \tag{A.11}$$

The rate of decay is controlled by parameter τ. Nair and Hinton [2009] and Hinton [2010] claim that for logistic units, the resulting gradient is simply $q_{j,t} - \widetilde{p}$.

From the cross-entropy loss $\mathcal{L}\left(q_{j,t}, \widetilde{p}\right)$ between the mean activation $q_{j,t}$ and the target mean $\widetilde{p}$, the weight update for the batch can be computed:

$$\begin{aligned} \Delta w_{ij,t} &= -\eta \frac{\partial \mathcal{L}\left(q_{j,t}, \widetilde{p}\right)}{\partial w_{ij}} \\ &= -\eta \left(\frac{1-\widetilde{p}}{1-q_{j,t}} - \frac{\widetilde{p}}{q_{j,t}}\right) \frac{\partial q_{j,t}}{\partial w_{ij}} \\ &= -\eta \frac{q_{j,t} - \widetilde{p}}{q_{j,t}\left(1-q_{j,t}\right)} \left(\frac{1-\tau}{K} \sum_{k=1}^{K} \frac{\partial z_{jk}}{\partial w_{ij}}\right) \\ &= -\eta \frac{q_{j,t} - \widetilde{p}}{q_{j,t}\left(1-q_{j,t}\right)} \left(\frac{1-\tau}{K} \sum_{k=1}^{K} z_{jk}(1-z_{jk}) x_{ik}\right) \end{aligned} \tag{A.12}$$

Due to the averaging effect of $q_{j,t}$ across the mini-batch, this cannot be further simplified to $-\eta\left(q_{j,t} - \widetilde{p}\right) x_i$, as claimed, even if logistic units are used.

A.4 Inverse temperature pooling parameters

The inverse temperature pooling parameter $\beta_{s,j}$ is updated using the partial derivative of the pooled value $v_{s,j}$ with respect to $\beta_{s,j}$. Combining Equations 6.5 and 6.6, we get

$$v_{s,j} = \frac{\sum_{n=1}^{N} z_{jn} r_{s,n} \exp\left(\beta_{s,j} z_{jn}\right)}{\sum_{n=1}^{N} r_{s,n} \exp\left(\beta_{s,j} z_{jn}\right)}. \tag{A.13}$$

Applying the quotient rule $\left(\frac{x}{y}\right)' = \frac{x'y - y'x}{y^2}$,

$$\begin{aligned}
\frac{\partial v_{s,j}}{\partial \beta_{s,j}} &= \frac{\left(\sum_{n=1}^{N} z_{jn}^2 r_{s,n} \exp\left(\beta_{s,j} z_{jn}\right)\right)\left(\sum_{n=1}^{N} r_{s,n} \exp\left(\beta_{s,j} z_{jn}\right)\right) - \left(\sum_{n=1}^{N} z_{jn} r_{s,n} \exp\left(\beta_{s,j} z_{jn}\right)\right)^2}{\left(\sum_{n=1}^{N} r_{s,n} \exp\left(\beta_{s,j} z_{jn}\right)\right)^2} \\
&= \frac{\sum_{n=1}^{N} z_{jn}^2 r_{s,n} \exp\left(\beta_{s,j} z_{jn}\right)}{\sum_{n=1}^{N} r_{s,n} \exp\left(\beta_{s,j} z_{jn}\right)} - \left(\frac{\sum_{n=1}^{N} z_{jn} r_{s,n} \exp\left(\beta_{s,j} z_{jn}\right)}{\sum_{n=1}^{N} r_{s,n} \exp\left(\beta_{s,j} z_{jn}\right)}\right)^2 \\
&= \left(\sum_{n=1}^{N} z_{jn}^2 \theta_{s,jn}\right) - \left(\sum_{n=1}^{N} z_{jn} \theta_{s,jn}\right)^2.
\end{aligned} \tag{A.14}$$

A.5 Degeneracy pooling parameters

To update the soft mask parameter $\alpha_{s,n}$ for the local coding z_{jn}, the partial derivative of the pooled value $v_{s,j}$ with respect to $\alpha_{s,n}$ is computed. Taking Equation 6.6, and applying the quotient rule $\left(\frac{x}{y}\right)' = \frac{x'y - y'x}{y^2}$, we get

$$\begin{aligned}
\frac{\partial v_{s,j}}{\partial \alpha_{s,n}} &= b_{s,n} \exp\left(\beta_{s,j} z_{jn}\right) \left(\frac{z_{jn} \sum_{\hat{n}=1}^{N} \alpha_{s,\hat{n}} b_{s,\hat{n}} \exp\left(\beta_{s,j} z_{j\hat{n}}\right) - \sum_{\hat{n}=1}^{N} z_{j\hat{n}} \alpha_{s,\hat{n}} b_{s,\hat{n}} \exp\left(\beta_{s,j} z_{j\hat{n}}\right)}{\left(\sum_{\hat{n}=1}^{N} \alpha_{s,\hat{n}} b_{s,\hat{n}} \exp\left(\beta_{s,j} z_{j\hat{n}}\right)\right)^2}\right) \\
&= \frac{b_{s,n} \exp\left(\beta_{s,j} z_{jn}\right)\left(z_{jn} - v_{s,j}\right)}{\sum_{\hat{n}=1}^{N} \alpha_{s,\hat{n}} b_{s,\hat{n}} \exp\left(\beta_{s,j} z_{j\hat{n}}\right)} \\
&= \left(z_{jn} - v_{s,j}\right) \frac{b_{s,n} \exp\left(\beta_{s,j} z_{jn}\right)}{\sum_{\hat{n}=1}^{N} \alpha_{s,\hat{n}} b_{s,\hat{n}} \exp\left(\beta_{s,j} z_{j\hat{n}}\right)} \cdot \frac{\alpha_{s,n}}{\alpha_{s,n}} \\
&= \frac{1}{\alpha_{s,n}} \left(z_{jn} - v_{s,j}\right) \theta_{s,jn}.
\end{aligned} \tag{A.15}$$

Bibliography

Aizerman, M. A., Braverman, E. M., and Ronzonoér, L. I. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–83. 47

Avila, S., Thome, N., Cord, M., Valle, E., and Araújo, A. (2011). BOSSA: extended BoW formalism for image classification. In *International Conference on Image Processing (ICIP)*. 46

Avila, S., Thome, N., Cord, M., Valle, E., and Araújo, A. (2013). Pooling in image representation: the visual codeword point of view. *Computer Vision and Image Understanding*, 117(5):453–465. 46, 47, 139

Barlow, H. B. (1961). Possible principles underlying the transformations of sensory messages. In Rosenblith, W., editor, *Sensory Communication*, pages 217–234. MIT Press. 50

Barlow, H. B. (1972). Single units and sensation: a neuron doctrine for perceptual psychology? *Perception*, 1(4):371–394. 50

Barlow, H. B. (1989). Unsupervised learning. *Neural Computation*, 1(3):295–311. 18, 50, 104

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110:346–359. 38, 102

Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202. 19

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127. 3, 27, 28, 30, 34

Bengio, Y. and Delalleau, O. (2009). Justifying and generalizing contrastive divergence. *Neural Computation*, 21(6):1601–1621. 24

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2006). Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems (NIPS)*. 4, 27, 29, 30, 33

Bengio, Y. and LeCun, Y. (2007). Scaling learning algorithms towards AI. In Bottou, L., Chapelle, O., DeCoste, D., and Weston, J., editors, *Large-Scale Kernel Machines*. MIT Press. 15, 16

Bileschi, S., Riesenhuber, M., Poggio, T., Serre, T., and Wolf, L. (2007). Robust object recognition with cortex-like mechanisms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:411–426. 107

Blackwell, D. (1947). Conditional expectation and unbiased sequential estimation. *Annals of Mathematical Statistics*, 18:105–110. 26

Boiman, O., Shechtman, E., and Irani, M. (2008). In defense of nearest-neighbor based image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 119

Boos, W. and Vogel, D. (1997). Sparsely connected, hebbian networks with strikingly large storage capacities. *Neural Networks*, 10(4):671–682. 50

Bordes, A. (2010). *New Algorithms for Large-Scale Support Vector Machines*. PhD thesis, Université Pierre et Marie Curie. 47

Bordes, A., Bottou, L., and Gallinari, P. (2009). SGD-QN: Careful quasi-Newton stochastic gradient descent. *Journal of Machine Learning Research*, 10:1737–1754. 47

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Workshop on Computational Learning Theory (COLT)*, pages 144–152. 47

Boureau, Y., Bach, F., LeCun, Y., and Ponce, J. (2010a). Learning mid-level features for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 19, 36, 43, 44, 45, 102, 103, 105, 112, 116, 117, 118, 119, 120, 124, 136

Boureau, Y., Le Roux, N., Bach, F., Ponce, J., and LeCun, Y. (2011). Ask the locals: Multi-way local pooling for image recognition. In *International Conference on Computer Vision (ICCV)*. 43, 103, 117, 118, 122

Boureau, Y., Ponce, J., and LeCun, Y. (2010b). A theoretical analysis of feature pooling in vision algorithms. In *International Conference on Machine Learning (ICML)*. 45

Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294. 16

Burghouts, G. J. and Geusebroek, J. M. (2009). Performance evaluation of local colour invariants. *Computer Vision and Image Understanding*, 113:48–62. 39

Chatfield, K., Lempitsky, V., Vedaldi, A., and Zisserman, A. (2011). The devil is in the details: an evaluation of recent feature encoding methods. In *British Machine Vision Conference (BMVC)*, pages 1–12. 38

Chen, H. and Murray, A. F. (2003). Continuous restricted Boltzmann machine with an implementable training algorithm. *IEE Proceedings of Vision, Image and Signal Processing*, 150(3):153–158. 26

Cireşan, D. C., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3642 – 3649. 100

Collobert, R. and Bengio, S. (2004). Links between perceptrons, MLPs and SVMs. In *International Conference on Machine Learning (ICML)*. 16

Cortes, C. and Vapnik, V. N. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297. 47

Courville, A., Bergstra, J., and Bengio, Y. (2011). The spike and slab restricted Boltzmann machine. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 233–241. 27

Cybenko, G. (1989). Approximations by superpositions of sigmoidal functions. *Mathematics of Control, Signals, and Systems*, 2(4):303–314. 15

Dahl, G., Ranzato, M., rahman Mohamed, A., and Hinton, G. E. (2010). Phone recognition with the mean-covariance restricted Boltzmann machine. In *Advances in Neural Information Processing Systems (NIPS)*, pages 469–477. 27

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4, 38, 102, 120

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 5

Deng, L. and Yu, D. (2011). Deep convex net: A scalable architecture for speech pattern classification. In *Interspeech*. 30, 100

Desjardins, G., Courville, A., Bengio, Y., Vincent, P., and Delalleau, O. (2010). Tempered Markov Chain Monte Carlo for training of restricted Boltzmann machines. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 145–152. 26

Doi, E., Inui, T., Lee, T.-W., Wachtler, T., and Sejnowski, T. J. (2003). Spatiochromatic receptive field properties derived from information-theoretic analyses of cone mosaic responses to natural scenes. *Neural Computation*, 15:397–417. 65

Doi, E. and Lewicki, M. S. (2005). Sparse coding of natural images using an overcomplete set of limited capacity units. In *Advances in Neural Information Processing Systems (NIPS)*, pages 377–384. 66

Duchenne, O., Joulin, A., and Ponce, J. (2011). A graph-matching kernel for object categorization. In *International Conference on Computer Vision (ICCV)*. 119, 125

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *Annals of Statistics*, 32(2):407–499. 19

Ekanadham, C. (2007). Sparse deep belief net models for visual area V2. Undergraduate Honors Thesis, Stanford University. 55, 147

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11:625–660. 29

Erhan, D., Manzagol, P.-A., Bengio, Y., Bengio, S., and Vincent, P. (2009). The difficulty of training deep architectures and the effect of unsupervised pre-training. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 153–160. 29

Fei-Fei, L., Fergus, R., and Perona, P. (2004). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Pattern Recognition Workshop*, 12:178. 6, 36, 48, 113, 134

Fei-Fei, L. and Perona, P. (2005). A bayesian hierarchical model for learning natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 113

Feng, J., Ni, B., Tian, Q., and Yan, S. (2011). Geometric ℓ_p-norm feature pooling for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 45, 119, 125

Földiák, P. (1991). Learning invariance from transformation sequences. *Neural Computation*, 3:194–200. 78

Földiák, P. (2009). Neural coding: non-local but explicit and conceptual. *Current Biology*, 19(19). 50, 51, 53, 54, 77

Földiák, P. (2013). Sparse and explicit neural coding. In Quiroga, R. Q. and Panzeri, S., editors, *Principles of Neural Coding*, pages 379–390. CRC Press. 50, 77

Földiák, P. and Young, M. P. (2002). Sparse coding in the primate cortex. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, Second Edition, pages 1064–4068. MIT Press. 51

Fournier, J., Cord, M., and Philipp-Foliguet, S. (2001). Retin: A content-based image indexing and retrieval system. *Pattern Analysis & Applications*, 4(2-3):153–173. 35, 42

Franco, L., Rolls, E. T., Aggelopoulos, N. C., and Jerez, J. M. (2007). Neuronal selectivity, population sparseness, and ergodicity in the inferior temporal visual cortex. *Biological Cybernetics*, 96(6):547–560. 51

Franz, M. O., Scholkopf, B., Mallot, H. A., and Bulthoff, H. H. (1998). Where did i take that snapshot? scene-based homing by image matching. *Biological Cybernetics*, 79:191–202. 37

Fu, W. J. (1998). Penalized regressions: The bridge versus the lasso. *Journal of Computational and Graphical Statistics*, 7:397–416. 19

Fukushima, K. (1980). A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):93–202. 33

Geusebroek, J., Burghouts, G., and Smeulders, A. (2005). The Amsterdam library of object images. *International Journal of Computer Vision*. 75

Goh, H., Kuśmierz, Ł., Lim, J.-H., Thome, N., and Cord, M. (2011). Learning invariant color features with sparse topographic restricted Boltzmann machines. In *International Conference on Image Processing (ICIP)*. 49, 144

Goh, H., Thome, N., and Cord, M. (2010a). Biasing restricted Boltzmann machines to

manipulate latent selectivity and sparsity. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*. 49, 144

Goh, H., Thome, N., Cord, M., and Lim, J.-H. (2010b). Neuroscience-informed sparsity and selectivity in restricted Boltzmann machines. Poster presented at *Decade of Mind VI Conference (DOM)*. 144

Goh, H., Thome, N., Cord, M., and Lim, J.-H. (2012). Unsupervised and supervised visual codes with restricted Boltzmann machines. In *European Conference on Computer Vision (ECCV)*. 101, 118, 144

Goh, H., Thome, N., Cord, M., and Lim, J.-H. (2013). Top-down regularization of deep belief networks. In *Advances in Neural Information Processing Systems (NIPS)*. 79, 144

Goh, H., Thome, N., Cord, M., and Lim, J.-H. (2014). Learning deep hierarchical visual feature coding. *IEEE Transactions on Neural Networks and Learning Systems*. 101, 118, 119, 144

Graham, B. and Willshaw, D. (1997). Capacity and information efficiency of the associative net. *Network: Computation in Neural Systems*, 8:35–54. 50

Grauman, K. and Darrell, T. (2005). The pyramid match kernel: discriminative classification with sets of image features. In *International Conference on Computer Vision (ICCV)*, volume 2, pages 1458 – 1465. 46

Griffin, G., Holub, A., and Perona, P. (2007). Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology. 6, 36, 113

Gross, C. G. (2002). Genealogy of the "grandmother cell". *Neuroscientist*, 8(5):512–518. 71

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800. 24

Hinton, G. E. (2007a). Learning multiple layers of representation. *Trends in Cognitive Sciences*, 11:428–434. 4

Hinton, G. E. (2007b). To recognize shapes, first learn to generate images. In Paul Cisek, T. D. and Kalaska, J. F., editors, *Computational Neuroscience: Theoretical Insights into Brain Function*, volume 165 of *Progress in Brain Research*, pages 535 – 547. Elsevier. 29

Hinton, G. E. (2010). A practical guide to training restricted Boltzmann machines.

Technical Report UTML TR 2010–003, Department of Computer Science, University of Toronto. 26, 27, 55, 147

Hinton, G. E., McClelland, J. L., and Rumelhart, D. E. (1986). Distributed representations. In Rumelhart, D. E., McClelland, J. L., and the PDP Research Group, editors, *Parallel Distributed Processing: Volume 1: Foundations*, chapter 3, pages 77–109. MIT Press, Cambridge. 12

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief networks. *Neural Computation*, 18(7):1527–1554. 4, 22, 27, 29, 30, 31, 80, 81, 82, 95, 96, 97, 98, 99, 107

Hinton, G. E., Peter, D., Frey, B. J., and Neal, R. M. (1995). The"wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161. 31, 81

Hinton, G. E. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 28:504–507. 22, 26, 30, 31

Hinton, G. E. and Sejnowski, T. (1986). Learning and relearning in Boltzmann machines. In Rumelhart, D. E., McClelland, J. L., and the PDP Research Group, editors, *Parallel Distributed Processing: Volume 1: Foundations*, chapter 7, pages 282–317. MIT Press, Cambridge. 20, 21

Hopfield, J. J. (1982.). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the USA*, 79(8):2554–2558. 21

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257. 3, 15, 27

Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *Journal of Physiology*, 160:106–154. 33

Hyvärinen, A. and Hoyer, P. O. (2001). A two-layer sparse coding model learns simple and complex cell receptive fields and topography from natural images. *Vision Research*, 41(18):2413–2423. 64

Hyvärinen, A., Hoyer, P. O., and Inki, M. (2001). Topographic independent component analysis. *Neural Computation*, 13(7):1527–1558. 73

Jégou, H., Douze, M., Schmid, C., and Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *IEEE Conference on Computer Vision and*

Pattern Recognition (CVPR). 42

Jia, Y., Huang, C., and Darrell, T. (2012). Beyond spatial pyramids: Receptive field learning for pooled image features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3370–3377. 137

Jiang, Z., Lin, Z., and Davis, L. S. (2011). Learning a discriminative dictionary for sparse coding via label consistent K-SVD. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 44, 118, 124

Jurie, F. and Triggs, B. (2005). Creating efficient codebooks for visual recognition. In *International Conference on Computer Vision (ICCV)*, pages 604–610. 38, 42

Karklin, Y. and Lewicki, M. (2008). Emergence of complex cell properties by learning to generalize in natural scenes. *Nature*. 28

Kavukcuoglu, K., Ranzato, M., Fergus, R., and LeCun, Y. (2009). Learning invariant features through topographic filter maps. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 73, 107, 119

Kavukcuoglu, K., Sermanet, P., Boureau, Y.-L., Gregor, K., Mathieu, M., and Cun, Y. L. (2010). Learning convolutional feature hierarchies for visual recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1090–1098. 19, 43

Ke, Y. and Sukthankar, R. (2004). PCA-SIFT: a more distinctive representation for local image descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 506–513. 40

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69. 42

Konorski, J. (1967). *Integrative activity of the brain: an interdisciplinary approach*. University of Chicago Press. 12, 52

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. 5, 34, 107

Kullback, S. and Leibler, R. (1951). On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86. 24

Larochelle, H. and Bengio, Y. (2008). Classification using discriminative restricted Boltzmann machines. In *International Conference on Machine Learning (ICML)*, pages

536–543. 22, 27, 81

Larochelle, H., Bengio, Y., Louradour, J., and Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, 10:1–40. 29, 33

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., and Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. In *International Conference on Machine Learning (ICML)*, pages 473–480. 33

Law, M., Thome, N., and Cord, M. (2012). Hybrid pooling fusion in the BoW pipeline. In *ECCV Workshop on Information Fusion in Computer Vision for Concept Recognition*. 38

Lazebnik, S. and Raginsky, M. (2009). Supervised learning of quantizer codebooks by information loss minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1294–1309. 44

Lazebnik, S., Schmid, C., and Ponce, J. (2004). Semi-local affine parts for object recognition. In *British Machine Vision Conference (BMVC)*, pages 779–788. 40

Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4, 6, 36, 46, 47, 48, 102, 112, 113, 116, 118, 128, 129, 134

Le, Q., Ngiam, J., Chen, Z., hao Chia, D. J., Koh, P. W., and Ng, A. (2010). Tiled convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1279–1287. 35

Le, Q. V., Monga, R., Devin, M., Corrado, G., Chen, K., Ranzato, M., Dean, J., and Ng, A. Y. (2012). Building high-level features using large scale unsupervised learning. In *International Conference on Machine Learning (ICML)*. 5

Le Roux, N. and Bengio, Y. (2008). Representational power of restricted Boltzmann machines and deep belief networks. *Neural Computation*, 20(6):1631–1649. 30

Le Roux, N. and Bengio, Y. (2010). Deep belief networks are compact universal approximators. *Neural Computation*, 22(8):2192–2207. 28

LeCun, Y. (1985). Une procédure d'apprentissage pour réseau a seuil asymmetrique (a learning scheme for asymmetric threshold networks). In *Cognitiva 85*, pages 599–604, Paris, France. 14

LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., and Jackel, L. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541 – 551. 5, 33, 34, 107

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. 5, 33, 34, 67, 71, 95, 107

Lee, H., Battle, A., Raina, R., and Ng, A. Y. (2007). Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, pages 801–808. 43

Lee, H., Ekanadham, C., and Ng, A. (2008). Sparse deep belief net model for visual area V2. In *Advances in Neural Information Processing Systems (NIPS)*, pages 873–880. 22, 27, 28, 44, 50, 54, 55, 63, 66, 107, 146

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *International Conference on Machine Learning (ICML)*, pages 609–616. 26, 27, 35, 44, 50, 107, 119

Leibe, B., Leonardis, A., and Schiele, B. (2008). Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1-3):259–289. 42

Levy, W. and Baxter, R. (1996). Energy-efficient neural codes. *Neural Computation*, 8(3):531–543. 50

Linde, Y., Buzo, A., and Gray, R. M. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28:84–94. 41

Liu, J. and Shah, M. (2007). Scene modeling using co-clustering. In *International Conference on Computer Vision (ICCV)*, pages 1–7. 42

Liu, L., Wang, L., and Liu, X. (2011). In defense of soft-assignment coding. In *International Conference on Computer Vision (ICCV)*. 42, 118

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137. 41

Lowe, D. (1999). Object recognition from local scale-invariant features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1150 –1157. 4, 36, 38, 39, 102, 116

Ma, W. and Manjunath, B. (1997). Netra: a toolbox for navigating large image databases. In *International Conference on Image Processing (ICIP)*, volume 1, pages 568–571. 35

Mairal, J. (2010). *Sparse Coding for Machine Learning, Image Processing and Computer Vision*. PhD thesis, Ecole Normale Superieure de Cachan. 19, 43

Mairal, J., Bach, F., Ponce, J., Sapiro, G., and Zisserman, A. (2008). Supervised dictionary learning. In *Advances in Neural Information Processing Systems (NIPS)*. 44

Marr, D. (1976). Early processing of visual information. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 275(942):483–519. 101

McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133. 12, 13

Memisevic, R. and Hinton, G. E. (2007). Unsupervised learning of image transformations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 27

Memisevic, R. and Hinton, G. E. (2010). Learning to represent spatial transformations with factored higher-order Boltzmann machines. *Neural Computation*, 22(6):1473–92. 27

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(27):1615–1630. 39, 40

Mitchell, T. M. (1980). The need for biases in learning generalizations. Technical Report CBM-TR-117, Departament of Computer Science, Rutgers University. 49

Mitchison, G. J. and Durbin, R. M. (1989). Bounds on the learning capacity of some multi-layer networks. *Biological Cybernetics*, 60(5):345–356. 50

Mohamed, A., Dahl, G. E., and Hinton, G. (2012). Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22. 5

Mutch, J. and Lowe, D. (2008). Object class recognition and localization using sparse features with limited receptive fields. *International Journal of Computer Vision*, 80:45–47. 107

Nair, V. and Hinton, G. E. (2009). 3d object recognition with deep belief nets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1339–1347. 27, 44, 50, 55, 63, 147

Neal, R. (1992). Connectionist learning of belief networks. *Artificial Intelligence*, 56(1):71–113. 26

Neal, R. M. and Hinton, G. E. (1998). A view of the em algorithm that justifies incremental, sparse, and other variants. In Jordan, M. I., editor, *Learning in graphical models*, pages 355–368. MIT Press, Cambridge, MA, USA. 82

Nister, D. and Stewenius, H. (2006). Scalable recognition with a vocabulary tree. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2161–2168. 42

Norouzi, M., Ranjbar, M., and Mori, G. (2009). Stacks of convolutional restricted Boltzmann machines for shift-invariant feature learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2735– 2742. 27, 35

Novikoff, A. B. (1962). On convergence proofs on perceptrons. In *Symposium on the Mathematical Theory of Automata*, pages 615–622. Polytechnic Institute of Brooklyn. 13

Nowak, E., Jurie, F., and Triggs, B. (2006). Sampling strategies for bag-of-features image classification. In *European Conference on Computer Vision (ECCV)*, pages 490–503. 38, 42

Oliva, A. and Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175. 37, 113

Oliveira, G., Nascimento, E., Vieira, A., and Campos, M. F. M. (2012). Sparse spatial coding: A novel approach for efficient and accurate object recognition. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2592–2598. 43

Olmos, A. and Kingdom, F. (2004). McGill calibrated colour image database. http://tabby.vision.mcgill.ca. 72

Olshausen, B. A. (2001). Sparse codes and spikes. In *Probabilistic Models of the Brain: Perception and Neural Function*, pages 257–272. MIT Press. 19

Olshausen, B. A. and Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607. 18, 50, 66

Perpiñán, C. and Hinton, G. E. (2005). On contrastive divergence learning. In *International Workshop on Artificial Intelligence and Statistics (AISTATS)*, pages 59–66. 24

Perronnin, F. and Dance, C. (2007). Fisher kernels on visual vocabularies for image categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. 42

Perronnin, F., Sánchez, J., and Mensink, T. (2010). Improving the Fisher Kernel for Large-Scale Image Classification. In *European Conference on Computer Vision (ECCV)*, pages 143–156. 47

Philbin, J., Chum, O., Isard, M., Sivic, J., and Zisserman, A. (2008). Lost in quantization: Improving particular object retrieval in large scale image databases. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1– 8. 42

Ponce, J., Berg, T., Everingham, M., Forsyth, D., Hebert, M., Lazebnik, S., Marszalek, M., Schmid, C., Russell, B., Torralba, A., Williams, C., Zhang, J., and Zisserman, A. (2006). Dataset issues in object recognition. In Ponce, J., Hebert, M., Schmid, C., and Zisserman, A., editors, *Toward Category-Level Object Recognition*, volume 4170 of *Lecture Notes in Computer Science (LNCS)*, pages 29–48. Springer-Verlag. 113, 137

Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. Y. (2007). Self-taught learning: Transfer learning from unlabeled data. In *International Conference on Machine Learning (ICML)*. 120, 124

Rakotomamonjy, A. (2013). Direct optimization of the dictionary learning problem. Submitted to *IEEE Transactions on Signal Processing*. 17

Ranzato, M., Boureau, Y., Chopra, S., and LeCun, Y. (2007). A unified energy-based framework for unsupervised learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 44

Ranzato, M. and Hinton, G. E. (2010). Modeling pixel means and covariances using factorized third-order Boltzmann machines. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2551–2558. 26

Ranzato, M., Mnih, V., and Hinton, G. E. (2010). Generating more realistic images using gated MRF's. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2002–2010. 35

Ranzato, M., Poultney, C., Chopra, S., and LeCun, Y. (2006). Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1137–1144. 5, 19, 20, 29, 30, 32, 34, 57, 66, 91

Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2:1019–1025. 107

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407. 26

Rolls, E. T. and Milward, T. (2000). A model of invariant object recognition in the visual system: Learning rules activation functions, lateral inhibition, and information-based performance measures. *Neural Computation*, 12:2547–2572. 78

Rolls, E. T. and Stringer, S. M. (2001). Invariant object recognition in the visual system with error correction and temporal difference learning. *Network: Computation in Neural Systems*, 12:111–129. 78

Rolls, E. T. and Treves, A. (1990). The relative advantage of sparse versus distributed encoding for associative neuronal networks in the brain. *Network: Computation in Neural Systems*, 1(4):407–421. 50, 104

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(386-408). 12

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323:533 – 536. 14

Salakhutdinov, R. and Hinton, G. E. (2007). Learning a nonlinear embedding by preserving class neighbourhood structure. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 100

Salakhutdinov, R. and Hinton, G. E. (2009). Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978. 22, 30

Salakhutdinov, R. and Hinton, G. E. (2012). An efficient learning procedure for deep Boltzmann machines. *Neural Computation*, 24(8):1967–2006. 30

Salakhutdinov, R., Mnih, A., and Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. In *International Conference on Machine Learning (ICML)*, pages 791–798. 22

Salton, G. and McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY, USA. 35

Schmid, C., Mohr, R., and Bauckhage, C. (2000). Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172. 38

Sejnowski, T. J. (1987). Higher-order Boltzmann machines. In *AIP Conference Proceedings 151: Neural Networks for Computing*, pages 398–403, Woodbury, NY. American Institute of Physics. 27

Sharma, G., Jurie, F., and Schmid, C. (2012). Discriminative spatial saliency for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3506–3513. 131

Sivic, J. and Zisserman, A. (2003). Video Google: A text retrieval approach to object matching in videos. In *International Conference on Computer Vision (ICCV)*. 35, 41

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In Rumelhart, D. E., McClelland, J. L., and the PDP Research Group, editors, *Parallel Distributed Processing: Volume 1: Foundations*, chapter 6, pages 194–281. MIT Press, Cambridge. 20, 22, 30, 44, 53, 102

Sohn, K., Jung, D. Y., Lee, H., and Hero III, A. (2011a). Efficient learning of sparse, distributed, convolutional feature representations for object recognition. In *International Conference on Computer Vision (ICCV)*. 44

Sohn, K., Jung, D. Y., Lee, H., and Hero III, A. (2011b). Efficient learning of sparse, distributed, convolutional feature representations for object recognition. In *International Conference on Computer Vision (ICCV)*. 118, 122, 125

Sutskever, I. and Hinton, G. E. (2007). Learning multilevel distributed representations for high-dimensional sequences. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 30

Sutskever, I. and Hinton, G. E. (2008). Deep, narrow sigmoid belief networks are universal approximators. *Neural Computation*, 20(11):2629–36. 28

Swain, M. J. and Ballard, D. H. (1990). Indexing via color histograms. In *International Conference on Computer Vision (ICCV)*, pages 390–393. 37

Swain, M. J. and Ballard, D. H. (1991). Color indexing. *International Journal of Computer Vision*, 7(1):11–32. 37

Swersky, K., Chen, B., Marlin, B., and de Freitas, N. (2010). A tutorial on stochastic approximation algorithms for training restricted Boltzmann machines and deep belief nets. In *Information Theory and Applications Workshop (ITA)*. 26

Szlam, A., Gregor, K., and LeCun, Y. (2012). Fast approximations to structured sparse coding and applications to object classification. In *European Conference on Computer*

Vision (ECCV 2012), pages 200–213. 19

Taylor, G. W. and Hinton, G. E. (2009). Factored conditional restricted Boltzmann machines for modeling motion style. In *International Conference on Machine Learning (ICML)*, pages 1025–1032. 27, 30

Teh, Y. W., Welling, M., Osindero, S., and Hinton, G. E. (2004). Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4(7-8):1235–1260. 66

Theriault, C., Thome, N., and Cord, M. (2013a). Dynamic scene classification: Learning motion descriptors with slow features analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 78

Theriault, C., Thome, N., and Cord, M. (2013b). Extended coding and pooling in the HMAX model. *IEEE Transaction on Image Processing*. 107

Tieleman, T. (2008). Training restricted Boltzmann machines using approximations to the likelihood gradient. In *International Conference on Machine Learning (ICML)*, pages 1064–1071. 26

Torralba, A. (2008). Small codes and large image databases for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. 37

Treves, A. and Rolls, E. T. (1991). What determines the capacity of autoassociative memories in the brain? *Network: Computation in Neural Systems*, 2(4):371–397. 68

Tuytelaars, T., Fritz, M., Saenko, K., and Darrell, T. (2011). The NBNN kernel. In *International Conference on Computer Vision (ICCV)*. 119

Tuytelaars, T. and Mikolajczyk, K. (2008). Local invariant feature detectors: a survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280. 37

van de Sande, K. E. A., Gevers, T., and Snoek, C. G. M. (2010). Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–1596. 39

van de Weijer, J. and Schmid, C. (2006). Coloring local feature extraction. In *European Conference on Computer Vision (ECCV)*, pages 334–348. 39

van Gemert, J., Veenman, C., Smeulders, A., and Geusebroek, J.-M. (2010). Visual word ambiguity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 40, 42, 47, 118

Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA. 3, 36, 47, 127

Vapnik, V. N. and Lerner, A. (1963). Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:778–780. 47

Vattani, A. (2011). *k*-means requires exponentially many iterations even in the plane. *Discrete and Computational Geometry*, 45(4):596–616. 42

Viitaniemi, V. and Laaksonen, J. (2008). Experiments on selection of codebooks for local image feature histograms. In *International Conference on Visual Information Systems (VISUAL)*, pages 126–137. 42

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning (ICML)*, pages 1096 – 1103. 16, 33

Wallis, G. and Rolls, E. T. (97). Invariant face and object recognition in the visual system. *Progress in Neurobiology*, 51(2):167–194. 78

Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., and Gong, Y. (2010). Locality-constrained linear coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 43, 103, 117, 118, 119, 120, 125

Welling, M., Osindero, S., and Hinton, G. E. (2003). Learning sparse topographic representations with products of student-t distributions. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1359–1366. 26, 73

Welling, M., Rosen-Zvi, M., and Hinton, G. E. (2005). Exponential family harmoniums with an application to information retrieval. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1481–1488. 26

Wellman, P. and Henrion, M. (1993). Explaining 'explaining away'. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):287–292. 30

Willmore, B. and Tolhurst, D. J. (2001). Characterizing the sparseness of neural codes. *Network: Computation in Neural Systems*, 12(3):255–270. 51, 53, 62

Wiskott, L. and Sejnowski, T. (2002). Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770. 78

Wolf, J., Burgard, W., and Burkhardt, H. (2002). Robust vision-based localization for mobile robots using an image retrieval system based on invariant features. In *IEEE*

International Conference on Robotics and Automation (ICRA). 37

Yang, J., Yu, K., Gong, Y., and Huang, T. (2009). Linear spatial pyramid matching using sparse coding for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1794– 1801. 43, 45, 47, 103, 117, 118, 120, 125, 134, 135

Yang, J., Yu, K., and Huang, T. (2010a). Efficient highly over-complete sparse coding using a mixture model. In *European Conference on Computer Vision (ECCV)*. 43

Yang, J., Yu, K., and Huang, T. (2010b). Supervised translation-invariant sparse coding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 44, 50

Yang, L., Jin, R., Sukthankar, R., and Jurie, F. (2008). Unifying discriminative visual codebook generation with classifier training for object category recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 44

Younes, L. (1989). Parametric inference for imperfectly observed Gibbsian fields. *Probability Theory and Related Fields*, 82(4):625– 645. 26

Younes, L. (1999). On the convergence of Markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics and Stochastic Reports*, 65(3-4):177–228. 26

Yu, K., Lin, Y., and Lafferty, J. D. (2011). Learning image representations from the pixel level via hierarchical sparse coding. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1713–1720. 107, 119

Yu, K., Zhang, T., and Gong, Y. (2009). Nonlinear learning using local coordinate coding. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2223–2231. 43

Zeiler, M. D. and Fergus, R. (2012). Differentiable pooling for hierarchical feature learning. *Computing Research Repository (CoRR)*, abs/1207.0151. 45

Zeiler, M. D. and Fergus, R. (2013). Stochastic pooling for regularization of deep convolutional neural networks. In *International Conference on Learning Representations (ICLR)*. 46

Zeiler, M. D., Krishnan, D., Taylor, G. W., and Fergus, R. (2010). Deconvolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2528–2535. 119

Zhang, H., Berg, A. C., Maire, M., and Malik, J. (2006). SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 119

Zhou, X., Yu, K., Zhang, T., and Huang, T. (2010). Image classification using super-vector coding of local image descriptors. In *European Conference on Computer Vision (ECCV)*. 42, 47

About the Author

Hanlin Goh graduated with a Ph.D. degree in computer science (with highest honours) from the Université Pierre et Marie Curie – Sorbonne Universités, Paris, France, in 2013. Prior to that, he received the B.Eng. degree in computer engineering and the M.Sc. degree in bioinformatics from the Nanyang Technological University, Singapore, in 2007 and 2009 respectively.

He is currently a Research Scientist with the Visual Computing Department at the *Institute for Infocomm Research, A*STAR*, Singapore, and a member of the *Image and Pervasive Access Laboratory, CNRS UMI 2955*, a French-Singaporean joint laboratory. His research motivation is to design machine learning algorithms that lead to scalable solutions for broad competence computer vision.

Dr. Goh was partially funded by the Merlion Ph.D. Scholarship Programme (bourse du gouvernement français), administered by the Embassy of France in Singapore, from 2010 to 2012.

Updated: July 2013

Contact details (France)

Laboratoire d'Informatique de Paris 6
Université Pierre et Marie Curie – Sorbonne Universités
4 Place Jussieu, 75005 Paris, France

☎ (+33) 1 44 27 51 29

✉ hanlin.goh@lip6.fr

🌐 http://webia.lip6.fr/∼gohh

Contact details (Singapore)

Institute for Infocomm Research
Agency for Science, Technology and Research
1 Fusionopolis Way, #21-01 Connexis (South Tower), Singapore 138632

☎ (+65) 6408 2491

✉ hlgoh@i2r.a-star.edu.sg

🌐 http://www1.i2r.a-star.edu.sg/∼hlgoh

« Dans la vie, rien n'est à craindre, tout est à comprendre. »

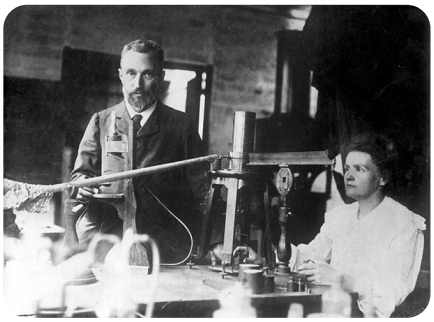
"Nothing in life is to be feared, it is only to be understood."

~ Marie Curie

« Il faut faire de la vie un rêve et faire d'un rêve une réalité. »

"Life should be made into a dream and a dream into a reality."

~ Pierre Curie



Pierre et Marie Curie, à Paris le 1898.

Learning Deep Visual Representations

Apprentissage de Représentations Visuelles Profondes