University of Alberta

LEARNING DEEP REPRESENTATIONS, EMBEDDINGS AND CODES FROM THE PIXEL LEVEL OF NATURAL AND MEDICAL IMAGES

by

Ryan Kiros

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Statistical Machine Learning

Department of Computing Science

Abstract

Significant research has gone into engineering representations that can identify high-level semantic structure in images, such as objects, people, events and scenes. Recently there has been a shift towards learning representations of images either on top of dense features or directly from the pixel level. These features are often learned in hierarchies using large amounts of unlabeled data with the goal of removing the need for hand-crafted representations.

In this thesis we consider the task of learning two specific types of image representations from standard size RGB images: a semi-supervised dense low-dimensional embedding and an unsupervised sparse binary code. We introduce a new algorithm called the deep matching pursuit network (DMP) that efficiently learns features layer-by-layer from the pixel level without the need for backpropagation fine tuning. The DMP network can be seen as a generalization of the single layer networks of Coates et. al. to multiple layers and larger images. We apply our features to several tasks including object detection, scene and event recognition, image auto-annotation and retrieval. For auto-annotation, we achieve competitive performance against methods that use 15 distinct hand-crafted features. We also apply our features for handwritten digit recognition on MNIST, achieving the best reported error when no distortions are used for training. When our features are combined with t-SNE, we obtain highly discriminative two dimensional image visualizations. Finally, we introduce the multi-scale DMP network for domain independent multimodal segmentation of medical images. We obtain the top performance on the MICCAI lung vessel segmentation (VESSEL12) competition and competitive performance on the MICCAI multimodal brain tumor segmentation (BRATS2012) challenge.

We conclude by discussing how the deep matching pursuit network can be applied to other modalities such as RGB-D images and spectrograms.

Acknowledgements

First and foremost I would like to thank my supervisor Csaba Szepesvári for his knowledgeable advice, kindness and flexibility in allowing me to explore a variety of topics of my interest. Our discussions have been invaluable to me and although we may not always agree, I have always left with additional insight, knowledge and new perspectives. I also would like to thank my committee members Russ Greiner and Linglong Kong for their feedback and constructive criticisms.

During my two years here I've gotten the opportunity to work and collaborate on many interesting projects. I would like to thank Dale Schuurmans for inviting me to work on his 2012 ICML dimensionality reduction project, as well as taking the time to respond to my many requests of letter writing and paper editing. I also thank Dana Cobzas and Martin Jagersand for their time and collaboration on our medical imaging projects, as well as the AICML for their generous support.

I am grateful for all the amazing friends and colleagues I've meet while living in Edmonton. You have all made this experience worthwhile.

Finally I would like my parents for their continuous love and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The use of machine learning in computer vision has led to several successful applications such as object detection, image segmentation, scene labeling and real-time recognition. These tasks require systems to identify higher order structure in images that go beyond the representation of pixel intensities. Furthermore, these representations have to be sufficiently useful for a classifier in order to learn the desired task from often large amounts of labeled examples.

We illustrate these points with a concrete example. Consider the examples of people in figure 1.1 [1]. Each image depicts a person with his/her right hand in the air, under various lighting conditions and background scenery. At a high level, one would arguably agree that these images are semantically similar due to the same pose that is exhibited in each image. Now consider the task of performing image retrieval, where our goal is to return semantically similar images from a given query [2]. Suppose that each image is represented as a collapsed vector of pixel intensities across RGB channels and retrieval is performed through Euclidean distance. Our retrieval system would perform poorly, since pixel distance cannot capture higher order semantics. As an example, a person standing in front of a blue background will be more likely to return images that also have a blue background but with other objects in front. Consequently, such representations would be of little use as input to a classifier, even if large amounts of labeled data were to exist for the desired task. Thus, an important task for successful deployment of recognition algorithms comes from the development of useful image representations.

Much work in past decades has gone into hand-crafting representations, that for example, could successful be used for our toy retrieval task in figure 1.1. Experts on vision and natural image

[1]Images taken from `https://www.ipam.ucla.edu/publications/gss2012/gss2012_10755.pdf`

[2]We discuss and experiment with retrieval further in Chapter 5.

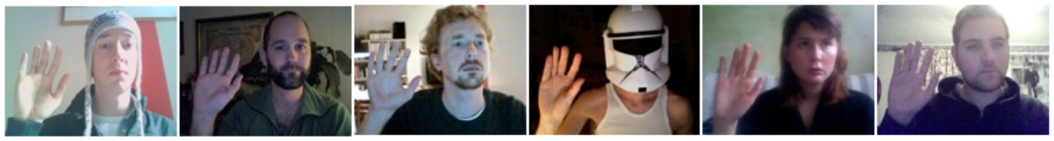


Figure 1.1: A depiction of 6 people raising their right hand. These images are semantically similar but have large Euclidean pixel distances between them.

statistics can import their prior knowledge into feature construction algorithms. Furthermore, these engineered features can be of a much lower dimension than the size of the image, allowing for improved scalability on learning tasks. Examples of hand-crafted features include SIFT [Lowe, 1999], HoG [Dalal and Triggs, 2005], SURF [Bay et al., 2006], Geometric Blur [Berg et al., 2005], RIFT [Lazebnik et al., 2004], Shape Context [Belongie et al., 2001] and many more. Given an image, one or more engineered descriptors can be computed from the image. The resulting output after processing will be a vector of fixed length. These vectors can then be used as input to a standard classifier, such as an SVM. At test time, descriptors for a new image are computed and a prediction is made from the trained classifier.

Over the past decade, many improvements and modifications have been introduced into this generic pipeline [Lazebnik et al., 2006, Bosch et al., 2007, Lampert et al., 2008, Yang et al., 2009, Wang et al., 2010]. Figure 1.2 illustrates a sample of these modifications. In a general setting, features are extracted from either interest points or densely across grids. An unsupervised learning algorithm, such as k-means, is applied to the features where centroids are then used to form a codebook. A histogram is computed from the codebook entries which serve as means of computing a specialized kernel, such as chi-squared or an intersection. These can then be used for training an SVM. Modifications to this standard pipeline include the use of spatial pyramid matching [Lazebnik et al., 2006] which aggregate features across multi-scale grids, as well as the use of sparse coding for quantization [Yang et al., 2009] leading to the effectiveness of linear spatial pyramid matching. As a result, features could be constructed that lead to high performance with linear classifiers, removing the extra computation demands of computing kernels.

Although this progression led to an improvement of recognition algorithms, what remained was the need to extract an initial representation, such as dense SIFT or HoG from the image which can be fed into further pipeline routines. This leads to several problems. Firstly, the choice of descriptor is largely task dependent. For example, HoG performs well on various pedestrian detection tasks [Dalal and Triggs, 2005], GIST [Oliva, 2005] is often used for scene recognition while

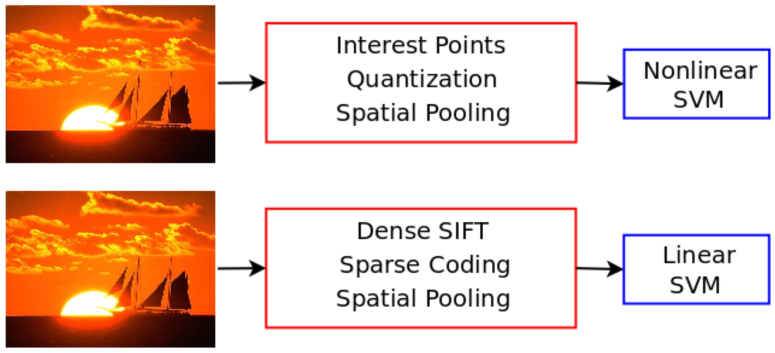


Figure 1.2: Two pipelines used for extracting and using features from images.

for image annotation, often a dozen or more descriptors are computed and aggregated [Guillaumin et al., 2009, Zhang et al., 2010]. Due to this, the choice of initial representation leads to a strong prior from which the remaining pipeline operations are to work with. Consequently, the choice of descriptor adds an extra dimension of experimentation when designing vision algorithms. Finally, it is often not clear how traditional descriptors can be used. An example of this is with RGB-D images, where depth information is included for most of the pixels.

These downsides lead to the alternative of instead learning useful features, as opposed to hand-crafting. In contrast to figure 1.2, we may feed an input image to a representation learning algorithm, which can output features appropriate to a classifier. If we reconsider our example of image retrieval in figure 1.1, a representation learning algorithm could instead learn an image representation that is more useful at identifying higher-order semantics of the image. Several other benefits exist in the usage of learning features, which are summarized as follows:

- **Use of unlabeled data.** Representation learning algorithms can employ the use of self-taught learning [Raina et al., 2007], which is a means of performing transfer learning on unlabeled data. For example, if our desired task is to construct a pedestrian detector, we can utilize random images off the internet, even those that may not include people in them, to build representation that improve pedestrian detection performance. Self-taught learning has been demonstrated to be especially effective in settings where only small amounts of labeled data are available [Raina et al., 2007, Lee et al., 2009a]. Throughout this thesis we reference learning features on different image distributions. By this we simply mean that features can be learned via self-taught learning on different objects and scenes that do not appear in the labeled training data.

- **Task applicability.** Learning features alleviates the need to not only experiment with hand-crafted features but also invent or modify new features for input images that are not grayscale or RGB. For example, existing representation learning algorithms are easily extended to RGB-D images [Blum et al., 2012, Bo et al., 2012a]. Furthermore, the focus can be shifted to better improving classifiers for the intended task.

- **Prior alleviation.** As previously mentioned, the choice of engineered features used introduces a strong prior on the remaining pipeline. As opposed to this, representation learning algorithms can harness the rich, high dimensional structure of an image and benefit from learning from low-level inputs. An extreme example of this is attempting to learn a short binary code to describe an image. A code construction algorithm could only work as well as the input descriptors. Alternatively, if code construction is combined with representation learning, the algorithm can make use of encoding semantic concepts learned directly from the pixel level.

Recently, much research has been invested into constructing deep, or hierarchical representation learning algorithms which are motivated by the successful approaches of training deep neural networks [Hinton et al., 2006, Bengio et al., 2007, Ranzato et al., 2006]. Deep learning algorithms aim to learn hierarchical representations of inputs in a layer-by-layer fashion. Consider the example of learning feature hierarchies of face images in figure 1.3, [3] which are computed from a convolutional deep belief network [Lee et al., 2009a]. The first layer of the algorithm identifies edge-detectors (gabors), the second layer learns features of parts of faces while the third layer features can identify the whole face. In this setting, the algorithm is able to learn higher order semantic concepts at each layer. The new representations that are encoded in a hierarchical fashion perform better at facial recognition then using a single layer alone [Lee et al., 2009a, Huang et al., 2012]. A survey of deep learning algorithms applied to images is detailed in Chapter 2.

A consequence of recent feature learning algorithms is a focus on the importance of encoding and development of efficient dictionary learning algorithms [Coates and Ng, 2011a]. Furthermore, these approaches can be successful without the need for backpropagation fine-tuning. As a consequence, the final representation of an image is often of very high dimension but with no guarantee

[3]Images obtained from `https://www.ipam.ucla.edu/publications/gss2012/gss2012_10595.pdf`

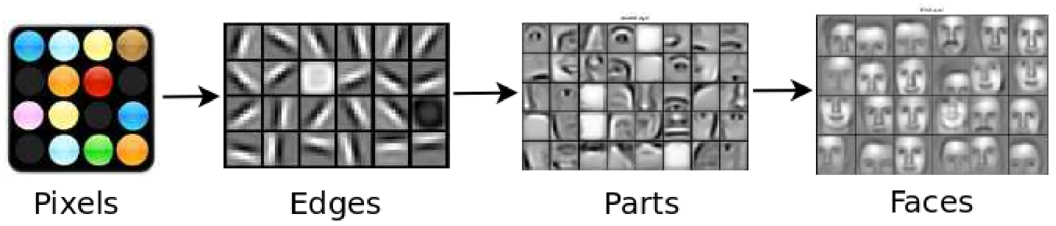


Figure 1.3: Sample features learned from a convolutional deep belief network. Each progressive layer extracts features that encompass higher level representations.

on its sparsity. While these representations have enjoyed good performance on a variety of recognition tasks, several vision tasks can benefit from compressed representations, such as our retrieval example in figure 1.

1.1 Thesis Contributions

The goal of this thesis is to consider the task of learning hierarchical embeddings and codes of full sized color and medical images directly from the pixel level. In particular, we develop algorithms for the following three tasks:

- Semi-supervised learning of dense low dimensional embeddings
- Unsupervised learning of sparse binary codes
- Learning voxel-based representations from medical images

where we focus on inexpensive dictionary learning algorithms that do not require the use of backpropagation to train. The first task of learning discriminative embeddings allows one to map an image to a low dimensional space (e.g. 50 dimensions) where the embeddings aim to be respective of object classes. The two main advantages of this setting is memory efficiency and visualization. Representing an image as a 50 dimensional real-valued vector allows for low memory consumption which can be applicable in settings where classification is performed on a large number of images. Finally, these representations are easily visualized in two or three dimensions using a visualization algorithm such as t-SNE [Van der Maaten and Hinton, 2008]. These low dimensional representations can be computed in a semi-supervised setting and make use of large amounts of unlabeled images to further improve performance.

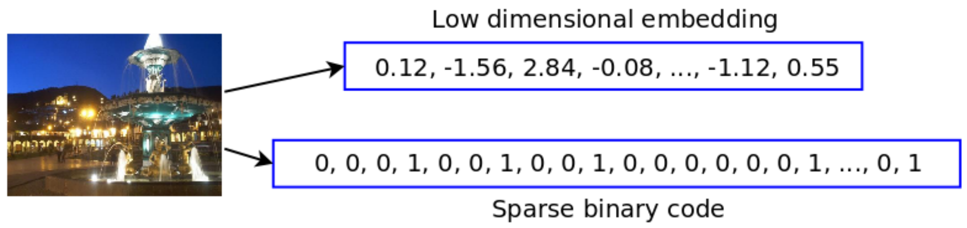


Figure 1.4: Two types of representations we focus on learning: semi-supervised, dense low dimensional embeddings and unsupervised sparse binary codes.

The second task involves the unsupervised learning of a mapping of a full-sized image to a short code (e.g. 256 bits). The learned codes are able to represent semantic concepts allowing the codes to be used for tasks such as the retrieval example in figure 1.1. As we show in Chapter 5, our codes can also be used in classification tasks where they are applied as a base representation for image auto-annotation. Finally, shorter codes (e.g. 30 bits) can be used to perform semantic hashing [Salakhutdinov and Hinton, 2009] on massive datasets, where retrieval is performed independent of the size of the database.

The final task, learning voxel-based representations from medical images, can be used to replace hand-crafted representations for segmentation algorithms. Our features can easily be integrated with variational methods and discriminative random fields, or used to train classifiers for classifying each voxel independently. In Chapter 6, we use our features on two medical imaging competitions: vessel segmentation in the lungs and multimodal brain tumor segmentation, where we achieve performances comparable or better to the current state of the art. Furthermore, our representations can be learned independent of the medical imaging domain used.

The algorithms used for performing the first two tasks are based on the premise of disentangling the factors of variation in high dimensional data [Bengio et al., 2012]. In a general setting, our approach of learning embeddings can be decoupled into two tasks. First, we construct a highly non-linear mapping of an image into a space where the recognition task can use linear classifiers. From there, simple existing linear mappings can be used to embed the data to the desired dimensionality. The principle is highly reminiscent of kernel learning and in particular we show that spectral embeddings fall into this setting. The non-linear mapping is parametrized in the form of a (possibly deep) convolutional architecture where the output representation combines encodings from multiple hierarchies. It is in this setting that the factors of variation as essentially ”disentangled” for which linear algorithms become applicable. Moreover, since these representations are

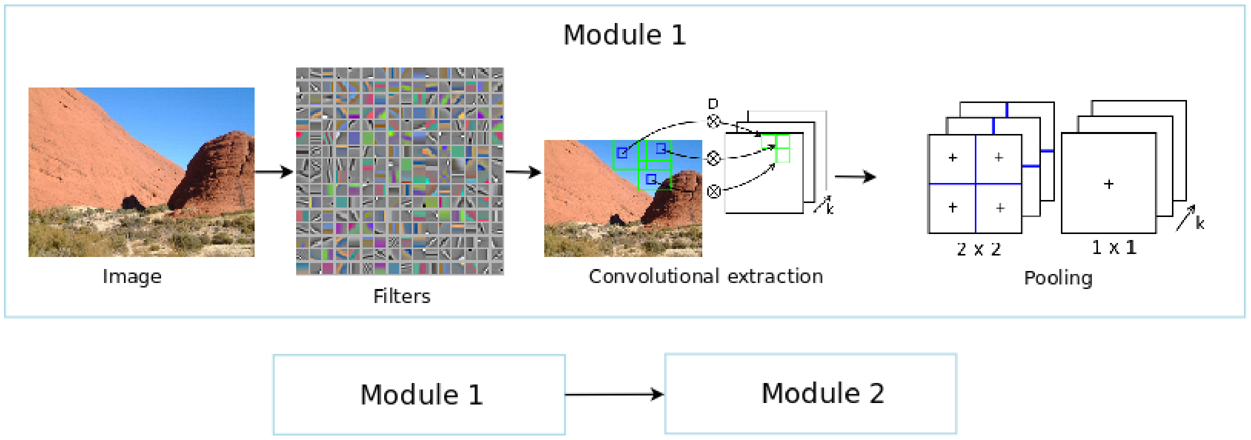


Figure 1.5: The deep matching persuit network described in Chapter 4. Each module consists of dictionary learning, convolutional feature extraction and pooling.

learned directly from the pixel level, no use of hand-crafted image descriptors is ever required. Finally, the networks described in this thesis can be trained unsupervised in a layer-by-layer fashion without any need for backpropagation fine-tuning.

The outline of this thesis is as follows:

- Chapter 2 gives a detailed summary of related deep learning, representation learning and dimensionality reduction algorithms.
- Chapter 3 studies the use of convolutional feature learning combined with linear mappings applied to tiny (thumbnail) images. We further relate the idea of disentangling the factors of variation to spectral embedding algorithms. Finally, experimental results of object detection are evaluated on two datasets: MNIST and CIFAR-10. We obtain competitive classification performance on CIFAR-10 with k-NN and 50 dimensional features compared to existing algorithms with tens of thousands of features. Using simple modifications to existing feature extraction algorithms, we obtain the best reported classification accuracy on MNIST when no additional image distortions are augmented to the training set.
- Chapter 4 introduces a new deep learning algorithm, which we call the deep matching pursuit network. DMP is able to learn multiple modules of features on full-sized color images. Each module consists of contrast normalization, dictionary learning, feature encoding and pooling phases. Benchmarking on the STL-10 datasets achieves state-of-the-art results. Furthermore,

we obtain similar embedding performance as our experiments in Chapter 3 but on full-sized images from the UIUC-Sports and MIT-Scenes datasets.

- Chapter 5 describes how the deep matching pursuit network can be used to learn short binary codes on full sized images. We apply our learned features to the task of image auto-annotation, where we achieve competitive results on the Natural Scenes, IAPRTC-12 and ESP-Game datasets. Furthermore, we compete with approaches that use over a dozen hand-crafted features, further illustrating the usefulness of learned feature hierarchies. Finally, we perform qualitative analysis of our codes for image retrieval, showing that our codes can capture high level semantic concepts of images.

- Chapter 6 introduces the multi-scale deep matching persuit network. The multi-scale DMP is used to learn pixel features, as opposed to global descriptors, by incorporating multiple scales and modules into the feature learning procedure. We apply the multi-scale DMP to two medical image segmentation challenges: vessel segmentation in the lungs (VESSEL12) and multimodal brain tumor segmentation (BRATS2012). We obtain the top result on VESSEL12 and competitive performance in the BRATS2012 competition.

- Chapter 7 concludes this work. In particular, we describe how simple modifications of our algorithms can allow them to be used for tasks beyond RGB images such as RGB-D, and audio. Finally, we discuss additional avenues of future research.

1.2 Publications and Public Research Challenges

This thesis is based on the following peer-reviewed papers:

- Ryan Kiros and Csaba Szepesvari. On Linear Embeddings and Unsupervised Feature Learning. Representation Learning Workshop, International Conference of Machine Learning (ICML), 2012. (Chapter 3)

- Ryan Kiros and Csaba Szepesvari. Deep Representations and Codes for Image Auto-Annotation. Proceedings in Neural Information Processing Systems (NIPS), 2012. (Chapters 4 and 5)

Other related peer-reviewed papers that do not appear in this thesis:

- Yaoliang Yu, James Neufeld, Ryan Kiros, Xinhua Zhang and Dale Schuurmans. Regularizers Versus Losses for Nonlinear Dimensionality Reduction. Proceedings in the International Conference on Machine Learning (ICML), 2012.

- Ryan Kiros. Training Neural Networks with Stochastic Hessian-Free Optimization. Proceedings in the International Conference on Learning Representations (ICLR), 2012.

The algorithms in this thesis were also used in public competitions:

- MICCAI Vessel Segmentation in the Lung , 2012. 1st place (out of 21). `http://vessel12.grand-challenge.org/`

- MICCAI Multi-modal brain tumor segmentation, 2012. Top 3 (out of 8, depending on the evaluation criteria). `http://www2.imm.dtu.dk/projects/BRATS2012/`

- Kaggle Gender Prediction from Handwriting, 2013. 7th place (out of 194). `http://www.kaggle.com/c/icdar2013-gender-prediction-from-handwriting`

- Kaggle Challenges in Representation Learning: Multi-Modal Learning, 2013. 5th place (out of 26).
  `http://www.kaggle.com/c/challenges-in-representation-learning-multi-modal-learning`

- Kaggle Challenges in Representation Learning: Facial Expression Recognition Challenge, 2013. 6th place (out of 56).
  `http://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge`

Chapter 2

Background

2.1 Deep Learning and Unsupervised Feature Learning

Since the development of the backpropagation algorithm [Rumelhart et al., 1986], much research was spent on training neural networks with many hidden layers, otherwise known as deep networks. Unfortunately, backpropagation was only successful on training networks with one or two hidden layers [Bengio et al., 2007]. The lack of success in training deep architectures can be explained by the vanishing gradient problem and the existence of difficult to optimize pathological curvature [Martens, 2010]. In a deep network, gradients that are backpropagated are most effective on layers that are near the output. As the signal passes towards the first layers of the network, the gradient is decayed due to the successive products being computed at each layer. By the time the signal reaches the first layer, the update on the first layer weights is too small to be effective. Since the updates are most effective at the output, this has the effect of averaging the error across the network. In shallow networks with a single hidden layer, the vanishing gradients are not as drastic and so such networks can be effectively trained in practice.

The first successful attempt at training deep architectures was the convolutional network [Le Cun et al., 1990]. Convolutional networks differ from standard multilayer perceptrons in that two types of layers are used: a convolutional layer and an aggregation (or pooling) layer. A signal is propagated in a convolutional layer by convolving the input with a filter kernel followed by a nonlinear activation function. In this sense, convolution acts as a form of weight sharing. As opposed to fully connected connections between layers, connections are arranged spatially as a receptive field across the input. Weight sharing is biologically motivated [1] and allows one to encode spatial ar-

[1]More details regarding biological motivation can be found at `http://www.iro.umontreal.ca/~pift6266/H10/notes/lenet.html`

rangements in the input, making it particularly applicable to images. After convolution, the pooling layer aggregates responses over a small spatial region and uses this as input to the next layer. Typical forms of pooling are max, average and L_p (p-norm). Pooling also has the effect of hard coding translational invariance into the network. Training is done using standard backpropagation.

One of the criticisms of neural networks is the non-convexity of its loss with respect to the input. In the mid 1990s, Vapnik and Cortes developed kernel support vector machines [Cortes and Vapnik, 1995], which offered nonlinear classification with a convex loss. Moreover, kernel SVMs could be seen as special type of perceptron with one hidden layer where the first layer computes the kernel and the second layer computes an output based on the support vectors. Due to the black box nature of SVMs, research in neural networks quickly faded and was replaced by kernel machines, particularly due to the success of the kernel trick.

It was not until 2006 that a successful approach to training non-convolutional deep networks was developed. Hinton et al. [2006] showed that a sigmoid belief network with infinitely many layers was equivalent to a restricted Boltzmann machine (RBM). An RBM is a bipartite Markov random field with a layer of visible units and a layer of hidden units, both of which were Bernoulli distributed. The RBM is governed by an energy function whose joint, marginal and conditional distributions over the visible and hidden units are defined through a Boltzmann distribution. RBM training is done through contrastive divergence [Hinton, 2002] which gives an approximation to the log likelihood of the marginal distribution over inputs. More specifically, training is performed to minimize the difference between expectations over the data and model distributions. Once training is complete, inference is performed on the hidden units, which can serve as inputs to another RBM. This kind of stacking, under certain conditions, can be used to prove a lower bound on the variational log likelihood on the input distribution. The resulting trained model after RBM stacking is known as a deep belief network (DBN). The deep belief network differs from a sigmoid belief network in that the top layer is undirected. Generating from the model simply involves running block Gibbs sampling between the top two layers followed by top-down propagation through the remaining layers. Due to the bipartite connections between layers, inferring the states of the hidden (visible) given the visible (hidden) units can be done exactly.

Besides proving a variational lower bound, Hinton et al. [2006] showed that the weights learned from RBM stacking could be used to initalize the weights of a deep neural network. This is known as 'pre-training' the network. It is exactly this pre-training step that allowed backpropagation to be applied successfully in training deeper architectures. Shortly after this discovery, Bengio et al. [2007] generalized this procedure as greedy layer-by-layer training and showed that autoencoders with a single hidden layer could also serve as an alternative module to pre-training. Furthermore Ranzato et al. [2006] showed more generally how energy-based models could also be used in training deep networks.

These discoveries have led to a resurgence in neural networks. In just 6 years, deep learning researches have developed algorithms to attack several problems in vision, audio, speech, natural language processing and more generally any type of problem that involves learning from high dimensional raw sensory input. Furthermore, the pretraining procedure can be seen as a type of unsupervised learning. This led to the hypothesis that first modeling the marginal input distribution allows one to better model the conditional distribution of labels given inputs. In other words, generative learning can be used to improve discriminative learning. This has opened much research in using large amounts of unlabeled data to build a model which, when trained discriminatively works better than just discriminative training alone. In our framework, unsupervised model learning can be seen as learning a map from one representation to another, from which the learned representation works better for discriminative training.

Since the focus on this thesis is learning representations from standard sized images, we spend the next section reviewing recently proposed representation learning algorithms for images. This is followed by a review of parametric dimensionality reduction as well as its applications in deep learning.

2.2 Large Scale Pixel-Level Representation Learning

Developing algorithms that were scalable to standard-size images remained a challenging task after the first developments in deep learning, with the exception of convolutional networks. One of the first proposed algorithms to challenge this was the convolutional deep belief network (CDBN) [Lee et al., 2009a]. The CDBN could viewed as a deep belief network that utilized a convolutional restricted Boltzmann machine for layer-by-layer training. Training the CRBM was identical to contrastive divergence in an RBM with the exception that the hidden activations were computed across a localized receptive field with tied weights. Furthermore, sparsity was no longer optional and was required due to the overcompleteness of the feature maps. Lee et al. [2009a]

also modified the energy function to incorporate pooling units that were computed probabilisticly. The CRBM then could be stacked to train multiple layers and could incorporate both bottom-up and top-down information and was used for hierarchical probabilistic inference. The CDBN has since been extended to many applications including audio classification and face verification [Lee et al., 2009b, Huang et al., 2012]. It could also be used to learn features on top of dense low-level descriptors [Sohn et al., 2011]. Figure 1.3 shows sample features learned from different layers of a convolutional DBN.

One of the difficulties that arises with the CDBN is difficulty in training due to the sensitivity of several parameters such as learning rate, momentum and weight decay. Sparse coding algorithms, on the other hand, required tuning at most 2 parameters and dealt with L1 optimizations that could be solved using algorithms such as FISTA [Beck and Teboulle, 2009] and feature-sign [Lee et al., 2007], alternating optimization of the dictionary and coding matrices. Sparse coding could be used on extracted patches from an image or trained convolutionally. The shortcoming that arises is the need to solve an optimization problem to obtain codes for each new image, which can be tedious. Kavukcuoglu et al. [2010a] proposed Predictive Sparse Decomposition (PSD) and a convolutional variation [Kavukcuoglu et al., 2010b] that uses a learned feedfoward encoder in addition to the L1 sparse coding objective. A special case of this algorithm, denoted reconstruction ICA [Le et al., 2011a], was used as a means of replacing the orthonormality constraint of ICA with a reconstruction cost. A variation of this algorithm was used by Google [Le et al., 2011c] to train a deep network which could identify faces and cats with unsupervised learning alone. The idea of mixing and matching the dictionary learning algorithm and the encoder was explored in detail by Coates and Ng [2011a], who concluded that emphasis should be placed on the encoder phase. Our proposed algorithm in Chapter 4 may be seen as a generalization of this work to larger images and deeper networks.

Almost all representation learning algorithms can be categorized as one of three categories: encoders, decoders and encoder-decoder. For example, PCA can be viewed as an encoder-decoder network, a one layer autoencoder with a linear activation while convolutional networks are encoders. Zeiler et al. [2011] proposed the deconvolutional network that is a decoder. In particular, feature maps are learned as to minimize the reconstruction of an image from these feature maps. Deeper networks can be trained jointly by taking the feature maps from the top layer and adjusting all weights as to minimize the reconstruction from the input. Deconvolutional networks also use a differentiable form of max pooling [Zeiler and Fergus, 2012] for easy error propagation.

Even with the development of the above representation learning methods, SIFT + sparse coding and linear spatial pyramid matching [Yang et al., 2009] was still the dominant pipeline for recognition tasks. It was not until the development of Hierarchical sparse coding [Yu et al., 2011] that led to a pixel-based representation learning algorithm that could outperform SIFT + sparse coding pipelines. Hierarchical sparse coding was followed by Hierarchical matching pursuit (HMP) [Bo et al., 2011b]. HMP utilized K-SVD dictionary learning [Rubinstein et al., 2008] combined with an efficient encoder using batch tree orthogonal matching persuit. The codes learned in one layer where then used to train a second layer of dictionary learning and encoding. HMP, as well as its followup for RGB-D recognition [Bo et al., 2012b] led to many state-of-the-art results in recognition and to this date is the current best performer for many of these tasks. The followup version of HMP uses separate dictionary training on multiple channels, such as grayscale, RGB, depth and surface normals for RGB-D images as well as multi-layer pooling. The consequence of this training results in having output vectors with over a hundred thousand features.

Due to the flexibility of training and the number of parameters and pre-processing operations, one may ask which operations are most relevant for representation learning and which pipelines are the most successful. Jarrett et al. [2009] performed an analysis of several pipelines and architectures for object recognition, such as the activation function, encoder, pooling and local contrast normalization. Coates et al. [2011] also performed analysis on single layer networks using autoencoders, RBMs, k-means and Gaussian mixture models with or without whitening and varying stride lengths. The concluding results were the significance of whitening and localized contrast normalization. Other forms of pooling have also been utilized such as L_p [Sermanet et al., 2012].

While most representation learning algorithms for images utilize convolution in some form, Hinton et al. [2011] argue that convolutional networks are sub-optimal and that the goal should be equivariance as opposed to hardwired invariance. To address this, Hinton et al. [2011] proposed the transforming autoencoder and Zeiler and Fergus [2012] utlized a ”what”/”where” decomposition in their deconvolutional networks. Hinton et al. [2012] recently introduced dropout backpropagation which may be viewed as a type of model averaging and regularization in neural networks. During a forward pass, a percentage of inputs and hidden units are randomly zeroed out as a means of reducing dependency amongst neurons. This training is reminiscent of denoising autoencoders [Vincent et al., 2008]. Hinton et al. [2012] utilized dropout backpropagation to zero out units in a fully connected top layer of a convolutional network, leading to state-of-the-art results on the 1000-way ImageNet classification task. Dropout backpropagation acts as a strong regularizers and removes the need for early stopping when training neural networks.

Other representation learning algorithms adapted to full size images include the deep-gated MRF [Ranzato et al., 2011], which can be viewed as convolutionally applying a factored, 3rd order RBM with mean and covariance hidden units, while Le et al. [2011b] utilized a convolutional form of independent subspace analysis for action recognition in video.

2.3 Parametric Dimensionality Reduction and Metric Learning

Methods for learning embeddings of high dimensional data can be broken into two categories: parametric and non-parametric methods. Non-parametric methods minimize an objective to learn a direct embedding of the input, do not learn an explicit mapping and have a data-dependent number of model parameters. This makes their applicability to out-of-sample data difficult and requires an extension such as a Nystrom approximation [Bengio et al., 2004]. Methods that fall into this category include Isomap [Tenenbaum et al., 2000], LLE [Roweis and Saul, 2000], Sammon mapping [Sammon Jr, 1969], Semidefinite embedding [Weinberger and Saul, 2004], stochastic neighbour embedding [Hinton and Roweis, 2002] and t-SNE [Van der Maaten and Hinton, 2008]. Since our goal is to learn a semi-supervised embedding from images, this requires us to consider an explicit mapping that can be used for unseen test datapoints, which is parametric in the sense that the number of model parameters is independent of the data. Thus, in this review we focus on parametric methods as well as nonlinear parametric embeddings developed from the use of deep architectures.

The simplest parametric embedding can be constructed through a linear transformation of the data with a learned, low rank matrix. This is done through the use of a learned Mahalanobis metric, for which the learned distance corresponds to Euclidean distance under the linear transformation. This type of metric learning is typically motivated for use in settings where nearest neighbor classifiers are used, avoiding the difficulty in choosing an appropriate distance between datapoints. This idea was first developed through neighborhood components analysis (NCA) [Goldberger et al., 2004]. NCA learned a supervised transformation of the data to maximize a smooth, stochastic variant to leave one out nearest neighbor classification accuracy on the training set. Maximally collapsing metric learning (MCML) [Globerson and Roweis, 2006], constructed two distributions, one proportional to the labels of the training data and the other to their distances in the embedding space, learning a metric as to minimize the KL divergence between these distributions. Large-margin nearest neighbor [Weinberger et al., 2006] (LMNN) proposes a semidefinite program to preserve neighbors in the embedding space with a precomputed set of target neighbors in the input space using a hinge loss. Correlative matrix mapping (CMM) [Strickert et al., 2010] learned a metric that maximizes the correlation of the embedding distance with the Euclidean distances between indicator label vectors.

Conceivably, one could replace the linear mapping with a nonlinear map such as a neural network. An objective is computed at the network output and the weights are updated through backpropagation. Prior to the use of pretraining, attempts at learning nonlinear mappings like this failed [Salakhutdinov and Hinton, 2007b]. With RBM pretraining, several new methods have been proposed. The first of these, by Hinton and Salakhutdinov [2006] who utilized RBM pretraining to learn a deep autoencoder. After pretraining several layers, the encoder was unravelled using tied weights and finetuned using backpropagation. This became the first successful attempt at training autoencoders with more than one hidden layer in its encoding and decoding phase. The same pretraining steps were also used for developing nonlinear NCA and LMNN [Salakhutdinov and Hinton, 2007a, Min et al., 2009]. van der Maaten [2009] proposed a parametric extension to t-SNE that uses a pretrained network and the t-SNE objective, adapting a heavy tailed distribution for modeling latent space similarities due to the crowding problem [Van der Maaten and Hinton, 2008], which is discussed in more detail in Chapter 3. Modifications for NCA and MCML that use heavy tailed distributions and a nonlinear embedding were proposed by Min et al. [2010] and led to improve k-NN performance over the same algorithms that instead use a Gaussian for identifying neighbors in the latent space.

Another class of algorithms for learning embeddings are based on the use of siamese architectures [Bromley et al., 1993]. Siamese architectures are used when training is performed on pairs of datapoints instead of the points themselves. A copy of the network is made and pairs of datapoints are ran through the networks. A distance is computed under the network mapping and an objective is minimized based on assigned labels to each image pair. Hadsell et al. [2006] proposed dimensionality reduction through learning an invariant mapping (DrLIM), which utilized a siamese convolutional network and a hinge loss for manifold learning with images. The gradient of the objective was further backpropagated through the convolutional network to updated the parameterization. [Taylor et al.] introduced convolutional NCA regression that proposes a modification of NCA for k-NN regression with application to pose estimation.

Chapter 3

Linear Embeddings of Convolutional Representations

3.1 Introduction

The ability to pre-train deep neural networks using modules such as an autoencoder or restricted Boltzmann machine have allowed for the development of highly non-linear parametric mappings for dimensionality reduction. These embeddings are parametrized by the network weights at each layer and further fine-tuned with backpropagation from the embedding objective function. Such methods outperform their linear counterparts when k-NN classification is used in the embedding space.

Surprisingly, little research has been done for producing embeddings of images that were extracted in convolutional settings such as [Coates et al., 2011] and [Coates and Ng, 2011a]. Such methods that were based on simple dictionary construction procedures such as k-means and orthogonal matching pursuit allow for thousands of bases and consequently thousands of extracted features. While the bases activations may be sparse, the final representations produced are dense and not guaranteed to have many non-zero elements. While these features can be efficiently used for linear classification, storage can become expensive for large image databases. Furthermore, tasks such as image retrieval would benefit from a small representation due to the need to compute pairwise distances on often millions of images.

In this chapter we study the task of learning embeddings of images whose representations have been convolutionally extracted. In particular, we show that linear embeddings are effective in reducing the dimensionality of the learned features from similar pipelines to Coates and Ng [2011a]. Using 50-dimensional supervised mappings is sufficient to obtain k-NN classification performance that is competitive with a linear SVM and significantly more features. Moreover, we show that self-taught learning is effective in this setting by utilizing datasets whose images come from a different distribution then that of the target dataset. Experimentation is performed on two datasets: CIFAR-10 and MNIST. Using simple modifications to existing architecture pipelines, we obtain a classification error of 0.40% on MNIST, the best reported result without appending distortions to the training set. The main advantage to this 2-step pipeline of representation learning and embedding over deep neural networks such as van der Maaten [2009] and Min et al. [2010] is no backpropagation is needed. Thus, low dimensional representations can be obtained by training each module one at a time without the need to re-adjust the weights learned from lower operations in the pipeline.

The organization of this chapter is as follows. We first introduce the pipeline for dictionary learning and convolutional feature extraction followed by a discussion on intrinsic dimensionality of data and the crowding problem. Next, we introduce the embedding algorithms as well as the modifications made to latent space representations to remove the crowding problem. We give an additional motivation for our approach by observing that all spectral methods for dimensionality reduction fall into the same 2-step embedding procedure. This is followed by experimental results and conclusion.

3.2 Representation Extraction

Let $I = \{I^{(1)}, \ldots, I^{(m)}\}$ be a set of m training images of size $n_V \times n_H$ denoting the height and length of an image. Let $n = 3n_V n_H$ denote the dimensionality of a color image and $n = n_V n_H$ for a grayscale image. For simplicity, assume that each image in I is of the same dimension, although this need not be the case. Further assume that I contains either all color or all grayscale images. Our goal is to learn a mapping $\phi : \mathbb{R}^n \rightarrow \mathbb{R}^{d_R}$ parametrized by a learned dictionary with k bases. The output dimensionality d_R is equal to $f(k)$ for some function f. We will assume that f is linear in the number of feature aggregation operations performed from spatial pooling and is independent of the final desired low dimensional embedding. We note that the training images I need not be the same as the training images from the target datasets. In other words, we can apply self-taught learning where I is an arbitrary image database. The pipeline operations for training, which mostly follows that of Coates and Ng [2011a] is as follows:

1. Randomly extract m_P patches of size $r \times c$ across all images. Pre-process each patch individually for brightness and contrast normalization.

2. Construct feature mean matrix M across all patches. Apply ZCA-whitening to the patches to obtain a whitening transform W.

3. Use orthogonal matching persuit (OMP) to learn a sparse dictionary D from whitened patches.

Let $\theta = \{M, W, D\}$ be the constructed extraction parameters. Given an image, the mapping ϕ consists of the following operations:

1. Convolutionally extract features using θ and a receptive field size of $r \times c$. Encode the outputs using a soft-activation function.

2. Spatially pool the features using a 2 layer pyramid. Concatenate the resulting features into a final feature vector of dimension d_R.

In the following subsections we describe each of these operations in detail.

3.2.1 Patch Extraction and Normalization

Let $P = \{p^{(1)}, \ldots, p^{(m_P)}\}$ denote a set of m_P patches randomly extracted from I of dimensionality $n_P = 3rc$ for color images or $n_P = rc$ for grayscale images. Here a color patch refers to the $r \times c$ receptive field over all three channels. Normalization is performed using mean centering and regularized variance normalization:

$$p^{(j)} \leftarrow \frac{p^{(j)} - \mu(p^{(j)})}{\sqrt{\sigma(p^{(j)})^2 + \gamma},} \quad 1 \leq j \leq m_p, \tag{3.1}$$

where $\mu(p^{(j)}), \sigma(p^{(j)})$ is the mean and standard deviation respectively of patch $p^{(j)}$. We set $\gamma = 10$ as was done by Coates and Ng [2011a]. This parameter is scale-dependent and assumes each pixel intensity remains between 0 and 255.

3.2.2 Zero Components Analysis (ZCA) Whitening

Whitening is an operation that is used to decrease intensity correlation between neighboring pixels. More specifically, after whitening, patches have zero mean, $\sum_{i=1}^{m_P} p^{(i)} = 0$, and identity covariance,

Algorithm 1 ZCA Whitening

```
Require: : $m_P \times n_P$ matrix of row-wise patches, regularization parameter $\epsilon$
Ensure: : $n_P \times 1$ mean vector $M$, $n_P \times n_P$ whitening matrix $W$
    $M \leftarrow \mu(P)$
    $P \leftarrow P - \mu(P)$
    $C \leftarrow \text{Cov}(P)$
    $VDV^T \leftarrow \text{eig}(C)$
    $W \leftarrow V(\Delta(\delta(D) + \epsilon))^{-\frac{1}{2}}V^T$
```

$\frac{1}{m_P}\sum_{i=1}^{m_P} p^{(i)}(p^{(i)})^T = I$. We use ZCA whitening, as was done by Coates and Ng [2011a], whose details are summarized in Algorithm 1.

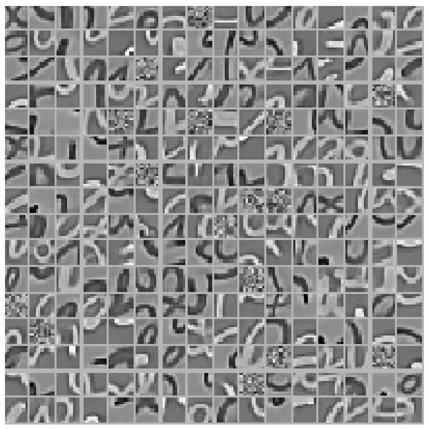
Let $P \in \mathbb{R}^{m_P \times n_P}$ be a $m_P \times n_P$ matrix where patches are assembled row-wise. We first compute a vector M of column (feature) means that are then subtracted column-wise from M. This is followed by computation of the covariance matrix C of P for which an eigendecomposition $P = VDV^T$ is calculated. The whitening matrix W is computed by scaling the eigenvalues λ_i by $\frac{1}{\sqrt{\lambda_i + \epsilon}}, i \in \{1 \ldots n_P\}$. The choice of ϵ has the effect of low-pass filtering the data if set sufficiently high. We use $\epsilon = 0.1$ for all experiments. The outputs M and W are used as part of the mapping parameters θ in order to apply whitening to unseen images.

3.2.3 Dictionary Construction

After performing whitening, the patches are now ready to be used for constructing a dictionary. We follow Coates and Ng [2011a] and use orthogonal matching persuit (OMP). OMP aims to find a solution to the following optimization problem (also referred to as spherical k-means):

$$\begin{aligned} \underset{D,\hat{p}^{(i)}}{\text{minimize}} \quad & \sum_{i=1}^{m_P} ||D\hat{p}^{(i)} - p^{(i)}||_2^2, \\ \text{subject to} \quad & ||D^{(j)}||_2^2 = 1, \forall j, \\ & ||\hat{p}^{(i)}||_0 \leq q, \forall i, \end{aligned} \tag{3.2}$$

where $D \in \mathbb{R}^{n_P \times k}$ and $D^{(j)}$ is the j-th column of D. Optimization is done using alternation over the dictionary D and codes $\hat{p}$. For all of our experiments we set $q = 1$, which reduces to a form of gain-shape vector quantization [Coates and Ng, 2011a]. In particular, given a dictionary D, an index k is chosen as


(a) MNIST

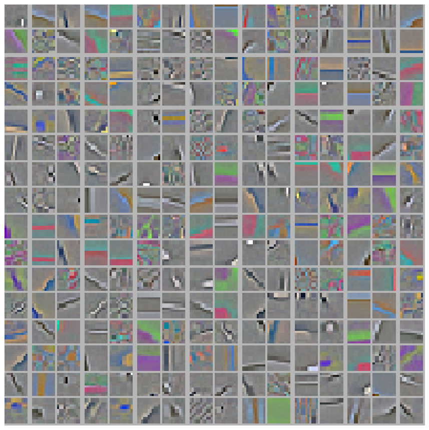

(b) STL-10

Figure 3.1: A random subset of bases learned using OMP on various datasets. On MNIST, bases correspond to various strokes and curves of digits while on the other datasets bases contain various edges and opponent colors.

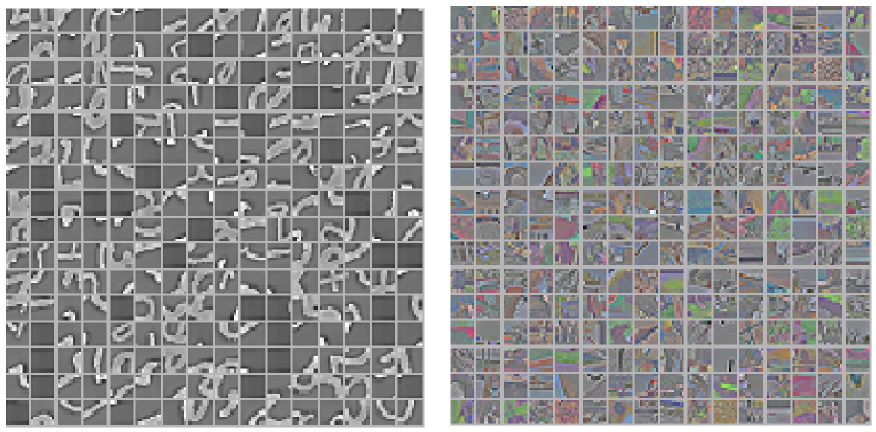
$$k = \underset{j}{\text{argmax}} |D^{(j)^T} p^{(i)}|, \tag{3.3}$$

for which the k-th index of $\hat{p}^{(i)}$ is set as $\hat{p}_k^{(i)} = D^{(k)^T} p^{(i)}$ with all other indicies set to zero in order to satisfy the constraint $\|\hat{p}^{(i)}\|_0 \leq 1$ for all i. Given the one-hot codes $[\hat{p}^{(1)}, \ldots, \hat{p}^{m_p}]$, the dictionary is easily updated and renormalized as to satisfy $\|D^{(j)}\|_2^2 = 1$ for all j. Figure 3.1 shows example bases learned using OMP.

Recently, Coates and Ng [2011a] showed that using randomly chosen patches as a dictionary can be surprisingly effective. Thus, we also consider constructing a dictionary by simply choosing a random subset of $\{p^{(1)}, \ldots, p^{(m)}\}$ to be the columns of D, followed by normalization with $\|D^{(j)}\|_2^2 = 1$. Zhang et al. [2011] showed that under a convex relaxation of sparse coding, global solutions can be obtained simply from normalizing over examples. This observation might help explain why such bases often perform well in practice.

3.2.4 Convolutional Feature Extraction

Let T denote an input image of size $n_V \times n_H$ with n_C channels and let T_j denote the j-th channel, $1 \leq j \leq n_C$, of T. For grayscale images, $n_C = 1$ and for color images $n_C = 3$. Each patch in T is pre-processed by contrast normalization, mean subtraction and whitening. Let $D_j^{(l)} \in \mathbb{R}^{r \times c}$ denote



(a) MNIST (b) STL-10

Figure 3.2: Random patch bases chosen from MNIST and CIFAR-10.

the l-th basis, $1 \leq l \leq k$, for channel j of D. We will define the feature encoding for basis l to be given by:

$$f_l = \max\Big\{\tanh\Big(\sum_{j=1}^{n_C} T_j * D_j^{(l)}\Big) - \alpha, 0\Big\}, \tag{3.4}$$

where * denotes convolution and α is a parameter to be set by the user. This type of surrogate encoding given by the rectification unit $max(x, 0)$ is motivated by Coates and Ng [2011a] and has been used in several algorithms including convolutional networks, restricted Boltzmann machines [Nair and Hinton, 2010] and deep sparse rectifier networks [Glorot et al., 2011]. The use of the tanh activation deviates from the pipeline of Coates and Ng [2011a] but we include it due to its successful use when evaluating multi-stage architectures for recognition [Jarrett et al., 2009].

The feature encoding f_l is of dimension $(n_V - r + 1) \times (n_H - c + 1)$. Let f denote the concatenation of the feature encodings f_l across bases l such that f has dimension $(n_V - r + 1) \times (n_H - c + 1) \times k$. Before pooling, we apply one additional form of local contrast normalization given by $\hat{f}^{(i)} \leftarrow (f^{(i)} - \mu(f^{(i)}))/\max\{\mu(\sigma), \sigma^{(i)}\}$ where μ and σ are means and standard deviations across patches, noting that $m_P = (n_V - r + 1)(n_H - c + 1)$. This type of normalization has been shown to be critical to performance by Jarrett et al. [2009] and Bo et al. [2011b]. The feature maps $\hat{f}_l$ are then used as inputs for pooling.

3.2.5 Pooling

Spatial pooling is performed by aggregating features over each encoding $\hat{f}_l$. This is done in the form of a 2 layer pyramid. The first layer sums over non-overlapping 4×4 spatial regions of $\hat{f}_l$ while the second layer sums over quadrants. After pooling over the first layer, the resuling output will be of dimension $\frac{(n_V - r+1)}{4} \times \frac{(n_H - c+1)}{4} \times k$, zero padding if necessary for divisibility. This region is then broken into quadrants and features are summed over each one. This results in a final feature vector of dimension $4 \times 4 \times k$. Prior to using these features for dimensionality reduction or classification, they are standardized to have zero mean and unit variance with the means and standard deviations preserved for application to unseen data.

3.3 Dimensionality Reduction

Let $X = \{x^{(1)}, \ldots, x^{(m)}\}, x^{(i)} \in \mathbb{R}^{d_R}, i = 1 \ldots m$ denote column vectors of standardized feature representations from the training data and let $l^{(i)}$ denote the associated label of $x^{(i)}$ given as a one-hot binary vector. For linear dimensionality reduction, our goal is to learn a mapping $\gamma : \mathbb{R}^{d_R} \rightarrow \mathbb{R}^{d_L}$ given by $\gamma(x^{(i)}) = Wx^{(i)}$ for some $W \in \mathbb{R}^{d_L \times d_R}$. We define the latent embedding space to be the d_L dimensional space for which the datapoints $\gamma(X)$ live, using the notation $\gamma(X)$ to indicate the embedding of all datapoints in X. From this point forward, we simply refer to this as the latent space.

3.3.1 Intrinsic Dimensionality and the Crowding Problem

It is commonly assumed under the manifold learning hypothesis [Cayton, 2005] that the data X, although represented as a d_R dimensional vector, actually lies on a manifold of a much smaller, unknown dimension d_I. We shall refer to d_I as the intrinsic dimensionality of X. Several algorithms exist for estimating the intrinsic dimensionality of data based on maximum likelihood estimates of the volume of a sphere centered around a datapoint [Levina and Bickel, 2004]. Furthermore, many algorithms have been proposed to learn this manifold itself, as described in Chapter 2. Alternatively, other algorithms aim to learn the local charts of a manifold such as the contractive autoencoder [Rifai et al., 2011a] and manifold tangent classifier [Rifai et al., 2011b].

When attempting to map X to some dimension d_R, the quality of the embedding depends highly on the relationship between d_R and d_I. If $d_R \geq d_I$, namely the latent space dimensionality is greater or equal to the intrinsic dimensionality, then the underlying manifold can be preserved. On the other hand, if $d_R < d_I$, then the mapping cannot naturally preserve the manifold. If we consider a d-dimensional sphere with radius r centered around a datapoint, then the volume of the sphere scales with r^d. Thus if we desire an embedding in a space less than its intrinsic dimensionality, (e.g. for data visualization), large pairwise distances between points in the original space have to be modeled as being further apart in the latent space. This leads to an issue known as the crowding problem described by Van der Maaten and Hinton [2008] and van der Maaten [2009]. Having larger pairwise distances in the latent space leads to small attractive forces between dissimilar points. The combination of these attractive forces effectively push all datapoints together and remove the clusters that were present between data in the original space. Modelling the latent similarities of data using a Gaussian distribution as was done with Stochastic Neighbor Embedding (SNE) [Hinton and Roweis, 2002] demonstrates the crowding problem. Note that if $d_R \geq d_I$, then the crowding problem does not exist.

The first attempt at removing the crowding problem was proposed with UNI-SNE [Cook et al., 2007], a variation of SNE, which involved introducing a repulsive force to counteract the attractive force between dissimilar points. Van der Maaten and Hinton [2008] pointed out that, although the approach was successful it resulted in a more difficult optimization problem then SNE. An alternative approach to removing the crowding problem was to utilize a heavy tailed distribution when modeling latent similarities between data. This was first used with t-SNE [Van der Maaten and Hinton, 2008] and later with parametric t-SNE [van der Maaten, 2009], dt-NCA and dt-MCML [Min et al., 2010] using a Student's t-distribution. t-SNE used two dimensional embeddings when a Student's t-distribution with one degree of freedom and later generalized with parametric t-SNE and to using $d - 1$ degrees of freedom for a d dimensional embedding. Alternately, one can attempt to learn the appropriate degrees of freedom through computing gradients with respect to the objective and updating the degrees of freedom after each iteration of the optimization [van der Maaten, 2009]. Note that when $d_R \geq d_I$ it is sensible to use a Gaussian for modeling latent space similarities since a Gaussian may be seen as a Student's t-distribution with infinite degrees of freedom.

In this chapter we consider embedding images into a 50 dimensional space, using variations of two existing Mahalanois metric learners: correlative matrix mapping [Strickert et al., 2010] and maximally collapsing metric learning [Globerson and Roweis, 2006]. Following parametric t-SNE, dt-CMM and dt-MCML, we model latent space similarities using a Student's t-distribution where the appropriate degrees of freedom are learned from the data.

3.3.2 Correlative Matrix Mapping

Correlative matrix mapping (CMM) earns a Mahalanobis type metric such that latent distances are in maximum correlation with label distances. Let D_L be a an $m \times m$ matrix whose i, j^{th} entry is the Euclidean distance between labels $l^{(i)}$ and $l^{(j)}$ of some given training data. Consider a Mahalanobis type distance of the form:

$$d_{ij} = \|\gamma(x^{(i)}; W) - \gamma(x^{(j)}; W)\|_2, \quad \gamma(x^{(i)}; W) = Wx^{(i)}, \tag{3.5}$$

between $x^{(i)}$ and $x^{(j)}$ for some $n \times d$ matrix W, where n is the number of features of $x^{(i)}$. CMM aims to learn W such that the pairwise label distances D_L and pairwise latent space distances D are in maximum correlation:

$$C = \max_W r(D_L, D) - \beta\|W\|_2^2, \tag{3.6}$$

where r denotes Pearson's correlation. The learned metric is equivalent to Euclidean distance with the transformed data $\gamma(x^{(i)}; W) = Wx^{(i)}$. The gradient may be expressed as:

$$\frac{\partial C}{\partial W} = \frac{\partial r(D_L, D)}{\partial D} \frac{\partial D}{\partial W}, \tag{3.7}$$

We now show how to modify CMM in order to model the latent space distributions with a Student's t-distribution. Let $\gamma(x^{(i)}; W) = Wx^{(i)}$ as before with $d_{ij} = \|\gamma(x^{(i)}; W) - \gamma(x^{(j)}; W)\|_2$. Consider a joint distribution Q with entries q_{ij} representing the probability that $x^{(i)}$ would have $x^{(j)}$ as a neighbor under a Student's t-distribution with α degrees of freedom:

$$q_{ij} = \frac{(1 + d_{ij}^2/\alpha)^{\frac{-\alpha+1}{2}}}{\sum_{k \neq l}(1 + d_{kl}^2/\alpha)^{\frac{-\alpha+1}{2}}}, \quad q_{ij} = 0, \tag{3.8}$$

Let D_L denote a joint distribution of the label similarities, where $(D_L)_{ij}$ is the cosine similarity between $l^{(i)}$ and $l^{(j)}$, normalized such that $\sum_{ij}(D_L)_{ij} = 1$. Our objective is now expressed as:

$$C = \max_W r(D_L, Q) - \beta\|W\|_2^2, \tag{3.9}$$

where Q is the joint distribution as defined. The degrees of freedom is initialized to $\alpha = d - 1$ and computing $\frac{\partial C}{\partial \alpha}$ at each update. The gradient of C with respect to W is given by:

$$\frac{\partial C}{\partial W} = \sum_i \frac{\partial C}{\partial \gamma(x^{(i)}; W)} \frac{\partial \gamma(x^{(i)}; W)}{\partial W}, \tag{3.10}$$

where the latter partial derivative is computed using standard backpropagation. We will denote the modified version of CMM by t-CMM.

3.3.3 Maximally Collapsing Metric Learning

The goal of MCML is to consider two distributions, one proportional to the label similarity of the training data and the other to the distances in the embedding space. The MCML objective is then to minimize the KL divergence between these two distributions. This may be seen as attempting to collapse all datapoints of the same class together, while mapping those of different classes to be ”infinitely” far apart. More specifically, consider the following conditional distribution with respect to datapoints $x^{(i)}$ and $x^{(j)}$:

$$q_{j|i} = \frac{\exp(d_{ij})}{\sum_{k \neq i} \exp(d_{ik})}, \quad i \neq j, \tag{3.11}$$

where $d_{ij} = (x^{(i)} - x^{(j)})^T W (x^{(i)} - x^{(j)})$ and W is positive semidefinite (PSD). Also consider the following conditional distribution over the label space:

$$p_{j|i} \propto \begin{cases} 1 & \text{if } y^{(i)} = y^{(j)}, \\ 0 & \text{if } y^{(i)} \neq y^{(j)}, \end{cases}$$

the MCML objective is to minimize the KL divergence between the densities P and Q:

$$C = \min_W \sum_i KL(P_i \| Q_i) = \sum_i \sum_{j \neq i} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

subject to the constraint that W is PSD. The optimization is done by randomly initializing a positive semidefinite matrix W_0 and performing gradient projection, where the projection is performed by computing an eigendecomposition of W and setting any negative eigenvalues to zero. The above formulation is convex in W and led to a dual and kernel formulation [Globerson and Roweis, 2006].

The modifications applied to MCML are straightforward, with some small changes. We replace the PSD matrix W with an unconstrained matrix and use the same distributions Q as was done for CMM. When a student's t-distribution is used, the objective is re-normalized as

$$C = \min_{W} KL(P\|Q) = \sum_{i}\sum_{j\neq i} p_{ij}\log\frac{p_{ij}}{q_{ij}} + \beta\|W\|_F^2, \tag{3.12}$$

We found that the use of regularization always improved performance in our experiments. We also note that our non-regularized MCML formulations can be seen as a special case of Min et al. [2010] when only a single layer with a linear activation is used for the embedding.

3.3.4 Optimization

For datasets that are sufficiently small, it is feasible to compute the objective and gradient on the full dataset and update using a standard optimization method such as L-BFGS. In the case of larger datasets, the pairwise objective makes it infeasible due to the requirement of computing pairwise distances amongst all datapoints. To account for this, we follow Salakhutdinov and Hinton [2007a] (among others) and use minibatch training. In particular, the training data is randomly partitioned into small batches of a few thousands datapoints. An iteration of the optimization is performed on each minibatches and an epoch consists of one pass of updates through each of the minibatches. Due to the non-convexity of the objective, we explored different optimization strategies for minibatch training. We found that L-BFGS with a sufficient amount of iterations per minibatch (we use 10 in our experiments) while maintaining early stopping on the number of epochs performed well in practice.

3.4 Motivation: Spectral Embeddings

Our combined procedure of feature learning with linear dimensionality reduction can be described as follows. We defined a mapping $\phi : \mathbb{R}^n \to \mathbb{R}^{d_R}$ parametrized by a learned dictionary D through a one layer convolutional feature extraction and pooling. The mapping ϕ is nonlinear due to the rectifier activation function. Next we introduced a mapping $\gamma : \mathbb{R}^{d_R} \to \mathbb{R}^{d_L}$ given by $\gamma(X) = WX$ for some learned matrix W. Thus, the composition of these operations, namely $(\gamma \circ \phi)(X)$, can be seen as a nonlinear embedding of an input image to a specified low dimensional space.

This type of representation learning is clearly reminiscent of kernel learning with the exception that the linear separable space mapped through ϕ is explicit as opposed to an implicit, perhaps infinite dimensional representation. Furthermore, our two step procedure for performing nonlinear dimensional reduction is also reminiscent of spectral methods for embeddings, such as kernel PCA and Semidefinite embedding. To see this, let $K \in \mathbb{R}^{m\times m}$ be a centered kernel matrix and let $D(K)$ denote the associated distance matrix. Consider the following objective:

$$\min_{\bar{K}=\bar{K}^T, K\succeq 0} L(D(\bar{K}); D(K)) + R(\bar{K}), \tag{3.13}$$

where L is a convex loss in its first argument and R a regularizer. Yu et al. [2012] showed that almost all non-parametric methods for dimensionality reduction can be expressed as a two step procedure: regularized loss minimization followed by eigenvalue truncation. Under this setup, let

$$\begin{aligned} \phi(K) = & \operatorname*{argmin}_{\bar{K}=\bar{K}^T, K\succeq 0} L(D(\bar{K}); D(K)) + R(\bar{K}) \\ \gamma(\bar{K}) = & \; Q_{:,1:d_L} V_{1:d_L,1:d_L} Q^T_{:,1:d_L}, \quad \bar{K} = QVQ^T \end{aligned} \tag{3.14}$$

where $\bar{K} = QVQ^T$ is an eigendecomposition of $\bar{K}$ and $V_{1:d_L,1:d_L}$ indicates the top d_L rows and columns of the matrix V sorted by eigenvalues. Observe that just as in our case, the mapping ϕ learns an appropriate representation for which an embedding is obtained through a linear operation. Put another way, the work in constructing a useful embedding comes in choosing an appropriate loss and kernel and is the differentiating factor between most spectral methods. Below we indicate how Kernel PCA and Semidefinite Embedding fall into this framework as described by Yu et al. [2012]:

- **Kernel PCA**: For kernel PCA, set $L(\hat{D}; D) = \|H(D - \hat{D})H\|$ and $R(\hat{K}) = [[rank(\hat{K}) \leq d_L]]$ where the notation $[[x]]$ is 0 if the predicate x is true and infinity otherwise.

- **Semidefinite Embedding**: For semidefinite embedding, otherwise known as MVU [Weinberger and Saul, 2004], we use $L(\hat{D}; D) = \sum_{i,j}[[N(D)_{ij} = 1 \text{ and } \hat{D}_{ij} \neq D_{ij}]]$ where $N(D_{ij})$ is 1 if datapoints $x^{(i)}$ and $x^{(j)}$ are neighbors and 0 otherwise. The regularizer used is $R(\hat{K}) = -tr(\hat{K})$.

The decomposition of nonlinear dimensionality reduction into a nonlinear representation component and linear embedding component gives great flexibility for representation learning. For

Figure 3.3: Sample training images from the CIFAR-10 dataset. Object classes correspond to columns.

example, when self-taught learning is utilized the learned dictionary D can be constructed on an entirely different database than the target dataset.

3.5 Experiments

We evaluate the performance of our proposed framework on two commonly used datasets in the deep learning / feature learning community: CIFAR-10 [1] [Krizhevsky, 2010] and MNIST [2]. MNIST is a collection of 70000 28×28 grayscale images of the digits 0-9. The dataset has a predetermined partition of 60000 digits for training and 10000 for testing. CIFAR-10 is a collection of 60000 32×32 color images with 10 classes. CIFAR-10 has a predetermined partition of 50000 images for training and 10000 images for testing.

For all experiments, we first apply unsupervised feature learning using the proposed architecture in Section 3.2, followed by learning a linear embedding using either t-CMM or t-MCML. On MNIST, we use two types of bases: learned using OMP $(\mathbf{P})$ and random patches $(\mathbf{RP})$ which are constructed from randomly selecting training patches and performing entrywise L2-normalization. On CIFAR-10 we use three types of bases: learned using OMP $(\mathbf{P})$, random patches $(\mathbf{RP})$ and

[1] `http://www.cs.toronto.edu/~kriz/cifar.html`

[2] `http://yann.lecun.com/exdb/mnist/`

bases learned using both CIFAR-10 and 100000 unlabeled images from STL-10 $(\mathbf{D})$. As an example, on CIFAR-10 we use both the unlabeled training set as well as the CIFAR-10 training set to learn a dictionary. We also consider semi-supervised experiments where 1%, 5% and 10% of labeled data is used. In this setting, only these percentages of data are used to learn the dictionary, while all the training data is used for unsupervised feature learning.

The procedure for experimentation is as follows. On all datasets, we learn a dictionary of size k = 4000 using 180000 patches for MNIST, 300000 patches for CIFAR-10. On all datasets, we use a receptive field size of 9×9. For learning the embeddings, we partition the training set into training and validation and apply either t-CMM or t-MCML until the objective value on the validation set does not improve for 10 epochs. Once this is completed, a value of k from $\{1, 3, 5, 10, 15, 20, 25\}$ is chosen that maximizes the classification accuracy on the validation set. The training and validation sets are then combined and the embedding algorithm is retrained for the chosen number of iterations. Classification accuracies are then reported using k-NN with the chosen k from the validation set. In all of our experiments, we fix the regularization parameter to $\beta = 0.01$ and soft-activation parameter $\alpha = 0.25$, which was used by Coates and Ng [2011a]

In some of our experiments we observed underfitting on the validation set, in the sense that although the objective value is no longer improving, the k-NN classification performance continued to improve. In this setting, we simply removed the early stopping criteria and ran the embedding algorithm for a total of 50 epochs. The best k is then chosen and the algorithm is retrained using the full training set for 50 epochs.

3.5.1 CIFAR-10

Experimental results for CIFAR-10 are shown in table 3.1. Notably, using additional bases learned from STL-10 along with the CIFAR-10 training data results in an improvement of results from just using the CIFAR-10 training data alone for unsupervised learning. Our best result of 80.1% accuracy, shown in table 3.2, is competitive with existing methods. This reinforces the principle of self-taught learning, that using unlabeled data even from a different distribution of the training data can lead to representations that can improve classification performance. We note that other results on CIFAR-10 exist based on the use of convolutional networks with backpropagation, namely [Krizhevsky, 2010, Cireşan et al., 2011, 2012]. The later is the current state-of-the-art, obtained through adding additional image transformations to the training data.

Table 3.1: k-NN classification accuracy of the learned 50 dimensional features on the CIFAR-10 dataset using various dictionaries and label percentages.

Method (% of labels)	P	RP	D
t-CMM (1%)	49.18%	47.27%	49.75%
t-MCML	50.25%	48.40%	50.46%
t-CMM (5%)	61.84%	60.95%	62.17%
t-MCML	64.01%	63.14%	64.31%
t-CMM (10%)	67.01%	66.39%	68.09%
t-MCML	68.72%	67.66%	69.68%
t-CMM (100%)	79.44%	78.21%	79.78%
t-MCML	78.69%	78.69%	**80.12%**

Table 3.2: A selection of the best results obtained on CIFAR-10 using non-backprop pipelines, sorted by number of features used.

Method	Accuracy	Features
OMP, k = 6000 [Coates and Ng, 2011a]	81.5%	48000
Receptive field learning [Jia et al., 2012]	83.1%	24000
Receptive field learning, 3 modules [Coates and Ng, 2011b]	82.0%	22400
k-means triangle, k = 4000[Coates et al., 2011]	79.6%	16000
Hierarchical kernel descriptors [Bo et al., 2011a]	80.0%	6000
1 module + SVM	80.0%	16000
1 module + t-MCML + k-NN	80.1%	50

Figure 3.4 visualizes the CIFAR-10 test data obtained using t-SNE on the 50 dimensional representations learned from MCML. A random subset of 2500 test data points were selected for training t-SNE. Interestingly, the result leads to two clusters: one containing man-made objects and the other containing living creatures.

3.5.2 MNIST

Table 3.4 describes our results obtained on MNIST. For additional comparison of our semi-supervised results, Lee et al. [2009a] obtain a result of 1.91% error with a convolutional deep belief network and Weston et al. [2008] a result of 1.83% using 5% of labelled data, where our best result of 1.05% significantly outperforms these methods.

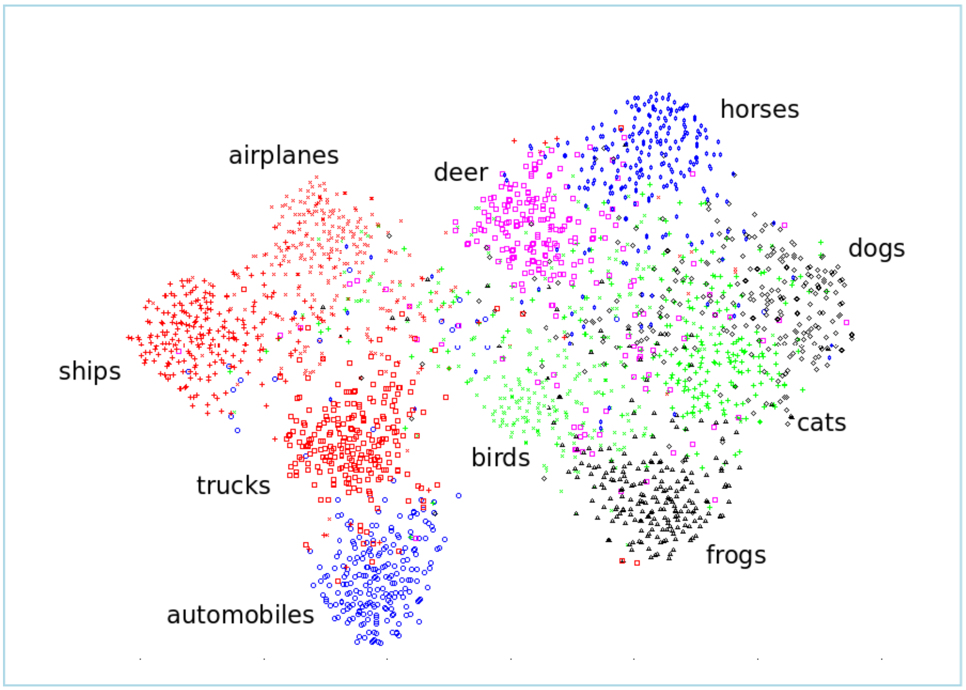


Figure 3.4: t-SNE embedding of the learned 50 dimensional features on a random subset of 2500 CIFAR-10 test datapoints. All the training data was used for learning both the dictionary and embedding (best seen in color)

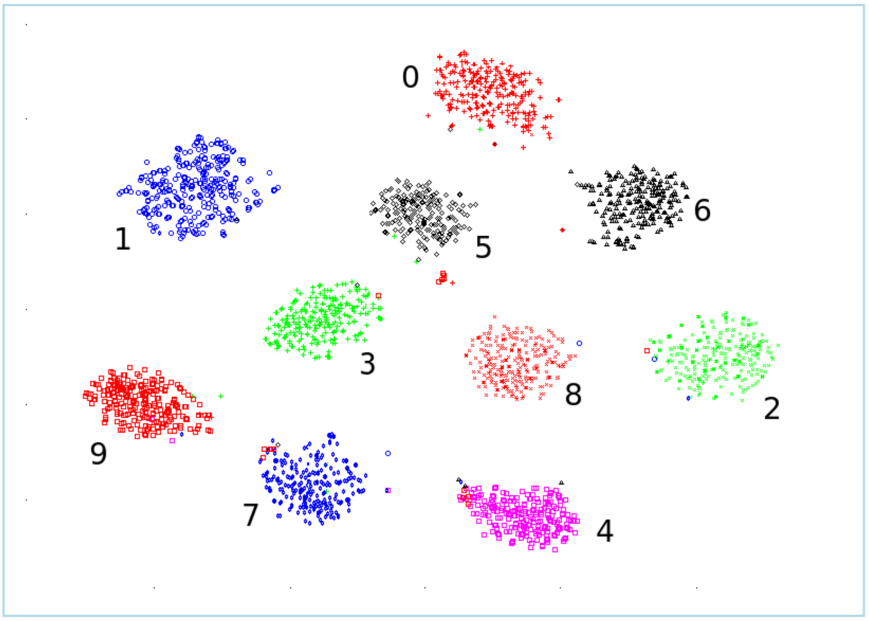


Figure 3.5: t-SNE embedding of the learned 50 dimensional features on a random subset of 2500 MNIST test datapoints, when only 1% of labeled training data was used to learn the t-CMM embedding. All the training data was used for unsupervised learning of the dictionary (best seen in color)

Table 3.4: A selection of the best results obtained on MNIST when no distortions are added to the training set.

Method	Error
Large conv. net, unsup features [Ranzato et al., 2007]	0.62%
Large conv. net, unsup pretraining [Ranzato et al., 2006]	0.60%
Unsupervised sparse features + SVM [Labusch et al., 2008]	0.59%
k-NN with non-linear deformation [Keysers et al., 2007]	0.54%
Large conv. net, unsup pretraining [Jarrett et al., 2009]	0.53%
k-NN with non-linear deformation [Keysers et al., 2007]	0.52%
Invariant scattering conv. net [Bruna and Mallat, 2012]	0.43%
1 module + t-CMM + k-NN	0.46%
1 module + t-MCML + k-NN	0.44%
1 module + SVM	0.40%

Table 3.4 compares our results with the best methods obtained on MNIST without the use of added distortions to the training set. To the best knowledge of the author, our error is the best reported, although error under 0.3% may be obtained with deep convolutional networks and image transformations [Cireşan et al., 2012]. [3]

Table 3.3: k-NN classification errors of the learned 50 dimensional features on the MNIST dataset using various dictionaries and label percentages.

Method (% of labels)	P	RP
t-CMM (1%)	2.51	2.98
t-MCML	2.60	2.74
t-CMM (5%)	1.05	1.23
t-MCML	1.06	1.06
t-CMM (10%)	0.79	0.87
t-MCML	0.93	0.80
t-CMM (100%)	0.46	0.52
t-MCML	**0.44**	0.51

Finally, as with CIFAR-10 we visualize a two dimensional embedding obtained from t-SNE on the 50 dimensional features from 2500 randomly selected test datapoints. Unlike CIFAR-10, we only use 1% of the training data for learning with MCML, while all the training data is used for unsupervised learning. Even with just 1% of labeled data, our features are able to naturally separate the digits. (Figure 3.5).

[3]Based on results posted on `http://yann.lecun.com/exdb/mnist/`

3.6 Conclusions

We have shown that simple linear embeddings applied on top of convolutional representations allow for classification performance that is competitive with a linear SVM and thousands more features. An important question for consideration is whether a Student's t-distribution is actually required for modeling latent space similarities. In other words, how critical is the crowding problem when the input features are linearly separable? We briefly performed experiments in the semi-supervised settings for all datasets when a Gaussian is used in the latent space. On MNIST, classification performance was comparable to that of the Student-t representations, while on CIFAR-10 classification performance was substantially worse. This shows that the intrinsic dimensionality of MNIST is lower than that of the 50-dimensional embedding space, in which case the crowding problem does not apply and infinite degrees of freedom can be used. Alternately, the intrinsic dimensionality of a color image would be much higher than that of a grayscale digit in which case 50 dimensions may be too low to successfully model the underlying latent manifold of the data. In this case, the crowding problem does apply and a heavy tailed distribution is required. Thus it is sensible to use a Student t-distribution whose degrees of freedom can be learned. For data with relatively low intrinsic dimension, the initial degrees of freedom is sufficient while on data with higher intrinsic dimension the degrees of freedom can be adjusted from the data.

An interesting avenue for future research could be to try to quantify the gap in classification accuracy between a linear SVM in the input space and that of a k-NN classifier in the embedding space with respect to the learned degrees of freedom from the Student's t-distribution. If the degrees of freedom remains high, then supposedly the data's intrinsic dimension is low and the gap will be small. Alternately if the degrees of freedom decreases, then the intrinsic dimensionality is high and the gap will be larger. Moreover, the learned degrees of freedom could be used to estimate the intrinsic dimensionality of the data's manifold.

Our experimental results in this chapter have been limited to tiny images. In the next chapter we introduce an architecture to learn feature representations from full sized color images. We perform identical embedding experiments on these representations and observe similar behavior in classification performance.

Chapter 4

Deep Matching Persuit Networks

4.1 Introduction

The feature extraction pipeline of Coates and Ng [2011a] with thousands of bases was shown to work surprisingly well on classification tasks for tiny images such as CIFAR-10. Furthermore, k-means feature learning was efficient and unlike sparse coding, the soft encoders of Coates et al. [2011] and Coates and Ng [2011a] do not require solving an optimization for each input image. It is natural to then ask whether or not this pipeline is easily extended not only into a deeper architecture but also scalable to more realistic sized images. As of current, two approaches based on learning receptive fields in higher layers were used to extend this pipeline beyond a single layer of feature learning.

Coates and Ng [2011b] showed that on tiny images, this pipeline could be extended beyond a single layer by incorporating a receptive field learning procedure. In early layers of a convolutional network, inputs have large spatial regions and few channels. For each new layer of feature learning, the number of bases in the previous layer becomes the number of channels in the next. Thus, with more than a couple of layers, the feature maps then have small spatial regions but a large number of channels. In this setting, a spatial receptive field is likely to take up most, if not all of the feature map. While engineering the receptive fields in lower layers is sufficient, in higher layers this is no longer useful. It is more sensible to instead consider receptive fields that encompass non-spatial features. Coates and Ng [2011b] randomly selected seeds from feature maps in upper layers and computed distances between features based on their squared correlations. Based on these distances, each feature is mapped to a receptive field indicated by its nearest seed. Separate dictionaries were then constructed based on each receptive field. With this additional procedure, Coates and Ng [2011b] was able to improve on a single layer on CIFAR-10 and 32×32 resolution STL-10 images. The second approach to receptive field learning considered was to use supervision to select which spatial regions to pool over. Jia et al. [2012] used grafting [Perkins et al., 2003] to select pooling regions to maximize classification accuracy. This approach led to a state-of-the-art result on CIFAR-10. This approach was also tested on the Caltech 101 dataset but used Dense SIFT for low level feature extraction.

The difficulty in performing extensive experimental evaluation on tiny images is that the results have no guarantee on generalizing to standard size images. Although receptive field learning was needed to construct a deep network on tiny images, it could simply be the case that multiple layers of feature extraction are not needed with such small images. In this chapter we introduce a new feature learning algorithm called the deep matching pursuit network. In particular, we scale the pipeline of Coates and Ng [2011a] to handle standard size images and show how multiple modules can be stacked, where a module is defined as a single pass of patch extraction, dictionary training, convolutional feature extraction and pooling. Interestingly, in all of our experiments, going beyond a single module always improves performance which is not the case in the experiments of Coates and Ng [2011b] on tiny images. Furthermore, we repeat the same batch of experiments that were performed in the previous chapter but on standard size images. Moreover, even on full-size images linear embeddings of multiple module representations are sufficient in constructing a highly discriminative low dimensional embedding. This shows that the deep matching pursuit network is able to learn a highly nonlinear mapping from pixel space that can be successfully fed into linear classifiers and embedding algorithms. Furthermore, no backpropagation is needed to fine-tune the model.

The rest of the chapter proceeds as follows. We introduce the deep matching pursuit algorithm by describing the operations performed in each module, which are largely similar to the pipeline in Chapter 3. We then show how multiple modules can be stacked to train a deep network. This is proceeded by experimental evaluation and finally conclusions.

4.2 Representation Extraction

We recall the definitions presented in Section 3.2. We assume we are given a list of training images $I = \{i^{(1)}, \dots, i^{(m)}\}$ each of dimension $n_V \times n_H$. Furthermore, in this chapter we assume all images are color, so that the dimensionality of an image is $n = 3n_V n_H$. Let $n_P = rc$ be the dimension of a receptive field of size $r \times c$ and $P \in \mathbb{R}^{m_P \times n_P}$ a matrix of m_P randomly extracted patches from I. Our goal is to learn a mapping $\phi : \mathbb{R}^n \to \mathbb{R}^{d_R}$ where d_R is the output dimensionality of the constructed feature representation. The mapping is parametrized by six matrices: mean matrices M_1 and M_2, whitening matrices W_1 and W_2 and dictionaries D_1 and D_2 from the first and second modules. The procedure for feature learning is similar to the single module networks of Chapter 3:

1. Randomly extract m_P patches of size $r \times c$ across all images. Pre-process each patch individually for brightness and contrast normalization.
2. Construct feature mean matrix M across all patches. Apply ZCA-whitening to the patches to obtain a whitening transform W.
3. Use K-SVD to learn a sparse dictionary D from whitened patches.
4. Convolutionally extract features and a receptive field size of $r \times c$. Encode the outputs using a soft-activation function
5. Spatially pool the features using a 2 layer pyramid.
6. Repeat steps 1-3 using a second module of training using the pooled inputs from step 5.

Let $\theta = \{M_1, W_1, D_1, M_2, W_2, D_2\}$ be the constructed extraction parameters. Given an image, the mapping ϕ consists of the following operations:

1. Extract, encode and pool features in the first module using the mean matrix M_1, whitening matrix W_1 and dictionary D_1.
2. Perform the same operations over a second module using M_2, W_2 and D_2. Concatenate the features extracted from both the first and second modules.

4.2.1 K-SVD Dictionary Learning

For dictionary learning, we follow Bo et al. [2011b, 2012a] and utilize the K-SVD algorithm [Rubinstein et al., 2008] [1]. K-SVD takes as input patches P and returns a dictionary D and sparse

[1]We use the implementation provided by `http://www.cs.technion.ac.il/~ronrubin/software.html`

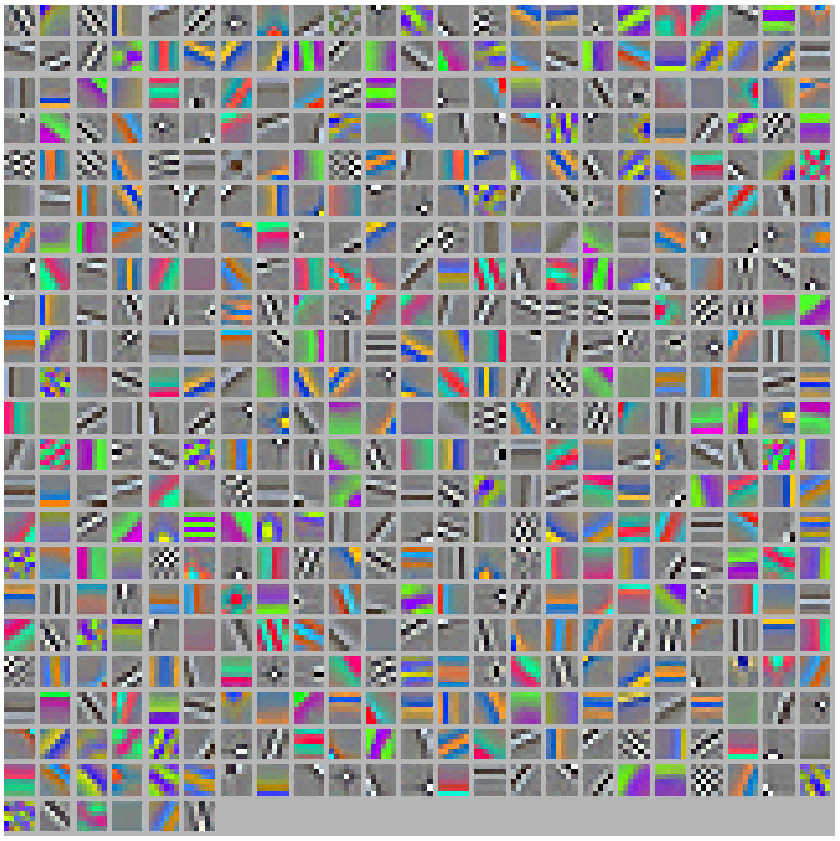


Figure 4.1: All 512 6×6 whitened bases learned from the Mirflickr dataset.

codes $\hat{P}$ by constructing a solution to the following optimization problem:

$$\underset{D,\hat{P}}{\text{minimize}} \quad \|P - D\hat{S}\|_F^2, \quad \text{subject to} \quad ||\hat{p}^{(i)}||_0 \leq q \;\; \forall i, \tag{4.1}$$

where $\hat{p}^{(i)}$ is the i-th column of P. Optimization is done in alternation: first fixing D and approximately solving for sparse codes $\hat{P}$, followed by fixing $\hat{P}$ and obtaining an updated dictionary D. When D is fixed, we may break the objective into m_P subproblems of the form $\|p^{(i)} - D\hat{p}^{(i)}\|^2$ subject to $||\hat{p}^{(i)}||_0 \leq q \; \forall i$ which can be approximately solved using batch orthogonal matching pursuit. For the case of $q = 1$, this reduces to the same problem as described in equation 3.2. Given the sparse codes $\hat{P}$ an updated dictionary is obtained by first re-writing the objective in terms of a residual $R^{(l)}, l = 1 \ldots k$:

$$\|P - D\hat{P}\|_F^2 = \|S - \sum_{j \neq l} d^{(j)} \hat{p}^{(j)^T} - d^{(l)} \hat{p}^{(l)^T}\|_F^2 = \|R^{(l)} - d^{(l)} \hat{s}^{(l)^T}\|_F^2, \tag{4.2}$$

A solution for the l-th column of D, namely $D^{(l)}$ is obtained through an SVD of $R^{(l)}$. The K-SVD algorithm may be seen as a generalization of k-means clustering. In particular, if $q = 1$ and P is forced to be binary, K-SVD reduces to k-means [Bo et al., 2011b].

4.2.2 Convolutional Feature Extraction

Let I be an input image and let T be a partition of tiles with each tile T_t having size $n_t \times n_t$ and stride length s between neighboring tiles. The patches in each T_t are pre-processed by contrast normalization, mean subtraction and whitening. Let T_{tl} denote the l-th channel for tile t and let $D_j^{(l)} \in \mathbb{R}^{r \times c}$ denote the j basis with channel l. The feature encoding f_{tl} for tile t and basis l is given by:

$$f_{tl} = \max\Big\{\tanh\Big(\sum_{j=1}^{n_C} T_{tj} * D_j^{(l)}\Big), 0\Big\}, \tag{4.3}$$

where * denotes a valid convolution and n_C is the number of channels. Note the similarity of this activation with that of the feature extraction done in Chapter 3. In particular, each tile T_t can be thought of what we denoted as an input image T. Thus, we are simply performing the same operation as before on each individual tile, although in this setting we set the soft-activation hyperparameter to $\alpha = 0$. After computing each f_{tl}, these feature maps are spatially concatenated into non-overlapping regions preserving the spatial locations of the tiles in the original image.

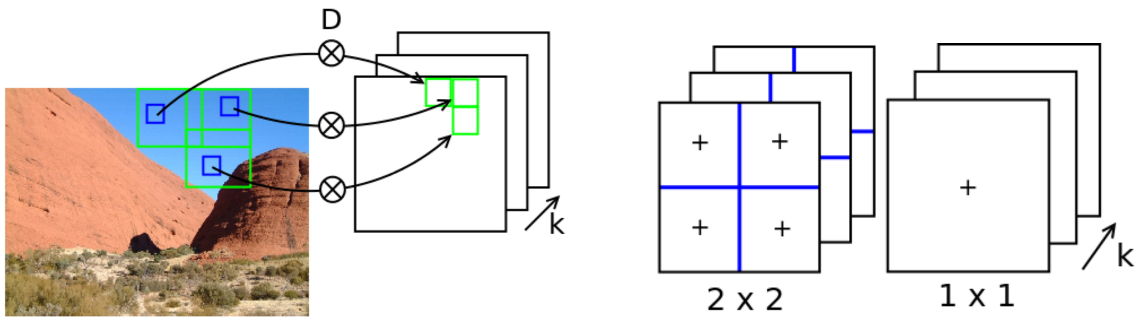


Figure 4.2: Left: D is convolved with each tile (large green square) with receptive field (small blue square) over a given stride. The outputs are re-assembled in non-overlapping regions preserving spatial structure. Right: 2×2 and 1×1 regions are summed (pooled) along each cross section.

See figure 4.2 for an illustration. Let f_t denote the concatenation of the feature encodings across bases. As was done before we perform an additional step of local contrast normalization given by $\hat{f}_t \leftarrow (f_t - \mu(f_t))/\max\{\mu(\sigma_t), \sigma_t^{(i)}\}$ where μ and σ are means and standard deviations across patches. Note that this is identical to the normalization done in Chapter 3 but here applied to each tile encoding separately.

4.2.3 Pooling

Let $\hat{f}$ denote the re-assembled encodings spatially organized by each encoded tile $\hat{f}_t$. Pooling is performed using a 2-layer pyramid but in this case we deviate from the pyramid in Chapter 2 due to the multi-resolution and sizes of the images. In particular, the first layer of the pyramid consists of summing across each basis cross section $\hat{f}_l$ over blocks of size 2×2 (quadrant pooling) while the top layer consists of summing all elements of $\hat{f}_l$. The bottom layer results in a feature vector of size $2 \times 2 \times k$ while the top layer is a vector of size $1 \times 1 \times k$. These are concatenated to form a final feature vector of size $4k + k = 5k$. Before being applied for classification or dimensionality reduction, the features are normalized to have zero mean and unit variance.

4.2.4 Training Multiple Modules

Recall that we defined a module as a single pass of patch extraction, whitening, dictionary learning, feature extraction and pooling. Up until now we described the procedure for performing each of the above in a single module. We can take the same operations and use them to train a second module using the outputs from the first. More specifically, each basis cross section $\hat{f}_l$ is partitioned into 256 blocks each of size 16×16 and pooling is performed over these blocks. When applied to each cross section, this results in an output of size $16 \times 16 \times k$. It is this output that we feed as input into a second module. For visualization, of second module training, refer to figure 4.2 but replace the image with the $16 \times 16 \times k$ output features. We chose this output size as a trade off between aggregating too much information (using a smaller output size) and not aggregating enough that would severely reduce the speed in training the second module.

More generally, multiple modules can be stacked in this fashion beyond using just two. We hypothesize that after two modules, receptive field learning is necessary in order to obtain further improvements.

4.3 Experiments

To test the effectiveness of our proposed feature learning algorithm, we apply the same set of experiments as was performed in Chapter 3 but on more realistic, standard size image datasets: STL-10 [Coates et al., 2011] [2], UIUC-Sports [Li and Fei-Fei, 2007] [3]and MIT Indoor Scenes [Quattoni and Torralba, 2009] [4]. For all experiments, we use k_1 = 512 first module bases, k_2 = 1024 second module bases, receptive field sizes of 6×6 and 2×2 and tile sizes (n_t) of 16×16 and 6×6. The first module stride length is chosen based on the length of the longest side of the image: 4 if the side is less than 128 pixels, 6 if less than 196 pixels and 8 otherwise. The second module stride length is fixed at 2. All these architecture parameters were hand-chosen on a development dataset not used for experiments. We report results using a linear L2-SVM for 1st module features as well as the concatenation of 1st and 2nd module features. Furthermore, we report k-NN classification accuracy of 50 dimensional embeddings of 1st and 2nd module concatenated features using t-CMM and t-MCML. Finally, for qualitative evaluation of our low dimensional features, t-SNE is used to visualize two dimensional embeddings on the test set learned from the 50 dimensional features, as was also done in Chapter 3.

Model selection is performed as follows. For SVMs, hyperparameters are chosen by 5-fold cross validation of the training set. For all embedding algorithms, optimization is done in batch mode, as opposed to mini-batch training in Chapter 3 since the training folds are sufficiently small.

[2] `http://www.stanford.edu/~acoates/stl10/`
[3] `http://vision.stanford.edu/lijiali/event_dataset/`
[4] `http://web.mit.edu/torralba/www/indoor.html`

During training t-CMM and t-MCML are optimized using L-BFGS for at most 100 epochs. 5-fold cross validation is used to select the regularization parameter β. We observed that the variance of the k-NN performance on the validation folds for various k was small enough to justify simply fixing k throughout all experiments, which we set to k = 10.

Self-taught learning is employed throughout our experiments. For the UIUC-Sports and MIT-Indoor Scenes datasets, feature learning is performed on the Mirflickr dataset [Huiskes and Lew, 2008] [5]. Mirflickr is a collection of 25000 images from Flickr that were voted to have high interestingness rating from users. Figure 4.3 illustrates a small sample of the dataset. Images contain a variety of objects, people, lighting conditions and poses. Thus, we hypothesize that Mirflickr offers a sufficient database to learn a general model for images. For feature learning, we randomly sampled 10000 images from Mirflickr. We used 500000 randomly sampled patches from these images to learn the first module bases and 250000 patches to learn the second module bases. The learned dictionaries and whitening matrices are then used for extracting features from UIUC-Sports and MIT Indoor Scene datasets. The STL-10 dataset comes with its own unlabeled dataset for self-taught learning which we describe in detail below.

4.3.1 STL-10

The STL-10 dataset is a collection of 10000 training images, 8000 testing images and 100000 unlabeled images, all of size 96×96. The training images are partitioned into 10 folds of size 1000 and contain 10 classes of objects. The procedure for evaluation on STL-10 is to apply self-taught learning on the unlabeled images, apply these feature maps to the training and testing images and finally report the average classification accuracy across all 10 folds. The images from the unlabeled dataset may or may not come from the same distribution (labeled classes) as those from training and testing.

Figure 4.1 shows a selection of the best results on STL-10. Learning second module features, in combination with the first module features, results in a statistically significant improvement over module alone ($p < 0.05$). This gives further evidence of the importance of training deeper architectures. Furthermore, k-NN performance in the 50 dimensional embedding space outperforms but is within the error bars of an SVM on first and second module features. Our approach outperforms all existing methods except for the recently proposed hierarchical matching pursuit algorithm (HMP)

[5] `http://press.liacs.nl/mirflickr/`

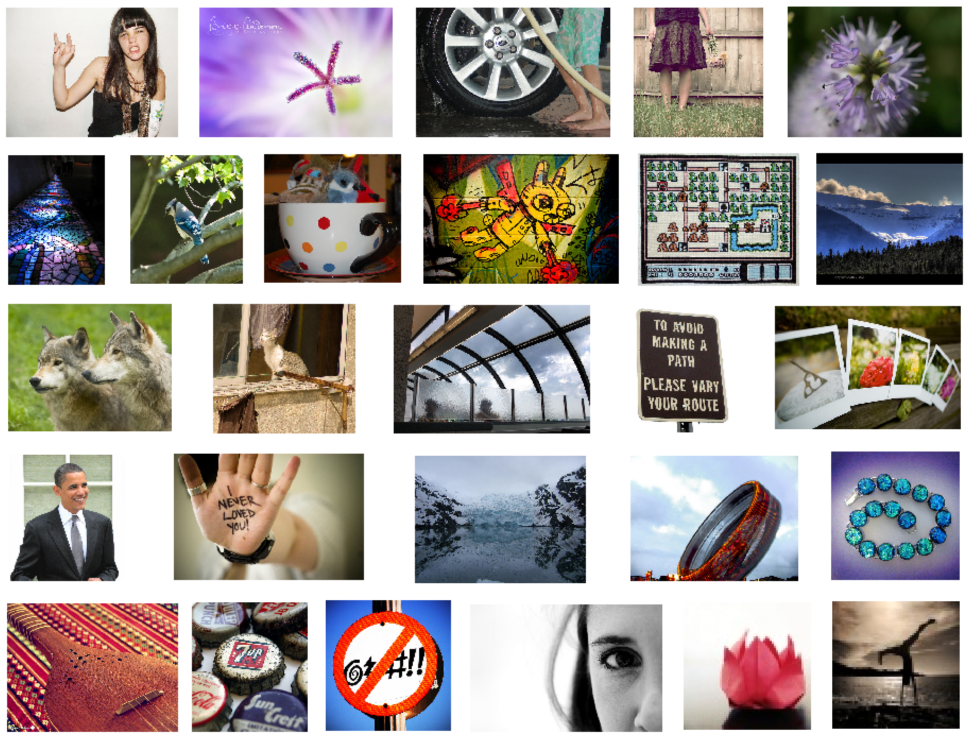


Figure 4.3: Sample images from the Mirflickr dataset, which is used for self-taught learning. Each image from Mirflickr was deemed to have a high interest rating.

[Bo et al., 2012a] which has much similarities to our proposed method. We note that unlike our training, HMP trains separate dictionaries on each channel and performs cross validation to select the architecture parameters. We hypothesize that further improvements can be made by performing the same procedures in our experiments, which Bo et al. [2012a] claim work better than training a single dictionary alone.

4.3.2 UIUC-Sports

UIUC-Sports is a collection of variable size images depicting activities from 8 sports: rowing, badminton, polo, bocce, snowboarding, croquet, sailing and rock climbing. Given an image, the goal is to accurately classify which sport is being played in the image. For feature extraction, we resized the images such that the longest side is at most 300 pixels with preserved aspect ratio. Evaluation is performed using a standard protocol from the paper of Li and Fei-Fei [2007]: 70 images per class are randomly sampled for training and 60 images per class for testing. This is repeated 10 times with the mean classification accuracy being reported.

Figure 4.2 describes the results of our algorithms in comparison to the state-of-the-art. As in

Table 4.1: A selection of the best results obtained on STL-10.

Method	Accuracy
k-means triangle, d = 4000 [Coates et al., 2011]	51.5%
Sparse filtering [Ngiam et al., 2011]	53.5%
OMP, $d = 1600$ [Coates and Ng, 2011a]	54.9%
OMP, SC encoder, $d = 1600$ [Coates and Ng, 2011a]	59.0%
Receptive field learning, 3 modules [Coates and Ng, 2011b]	60.1%
Video unsup features [Zou et al., 2011]	61.0%
Hierarchical matching pursuit, 2 modules [Bo et al., 2012a]	64.5%
1st module + SVM	56.4% (1.06%)
1st + 2nd module + SVM	62.1% (0.63%)
1st + 2nd module + t-CMM + k-NN	62.7% (0.81%)
1st + 2nd module + t-MCML + k-NN	62.7% (0.78%)

Table 4.2: A selection of the best results obtained on UIUC-Sports.

Method	Accuracy
SIFT + GGM [Li and Fei-Fei, 2007]	73.4%
Object bank [Li et al., 2010]	76.3%
SIFT + sparse coding [Bo et al., 2011b]	82.7%
MMLDA + SIFT + Gist [Wang and Mori, 2011]	83.1 %
Hierarchical matching pursuit, 2 modules [Bo et al., 2011b]	85.7%
1st module + SVM	78.3% (1.46%)
1st + 2nd module + SVM	83.8% (1.20%)
1st + 2nd module + t-MCML + k-NN	84.5% (1.13%)
1st + 2nd module + t-CMM + k-NN	84.6% (1.32%)

the case of STL-10, we achieve a statistically significant improvement in performance by training with 2 modules ($p < 0.05$). The learned features outperform more sophisticated algorithms, such as the the generative model of Li and Fei-Fei [2007] and descriptor combination with LDA model of Wang and Mori [2011]. Again, we outperform all existing approaches except for HMP. Figure 4.4 depicts a two dimensional visualization of test datapoints from one of the ten random train/test sample selections. Using the learned 50 dimensional embeddings, t-SNE is able to learn a discriminative embedding of the event classes.

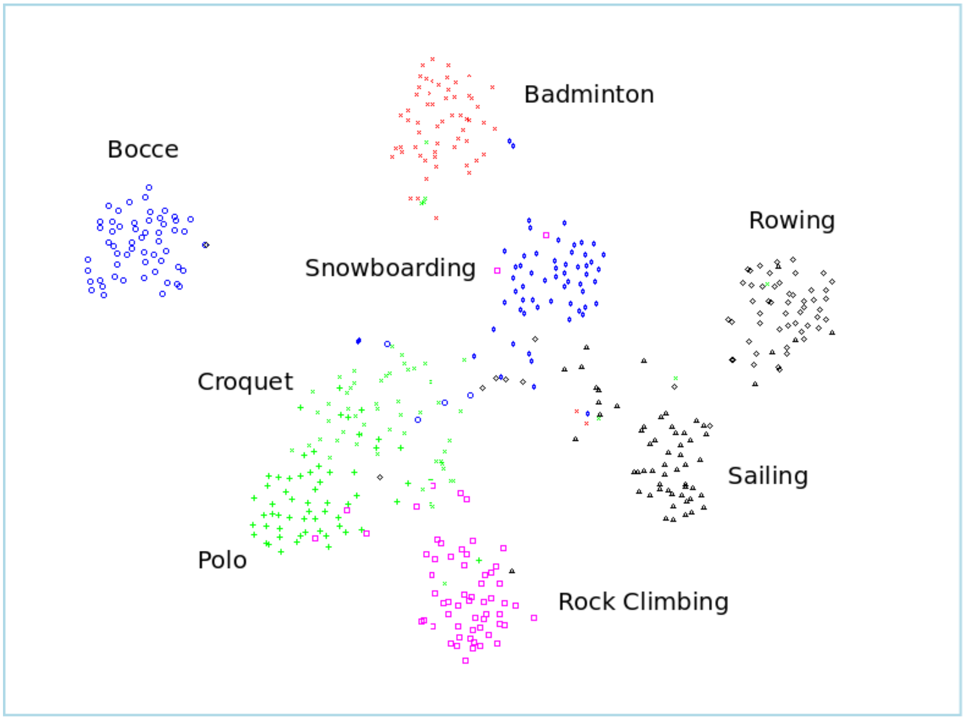


Figure 4.4: t-SNE embedding of the learned 50 dimensional features on one of the 10 UIUC test folds.

4.3.3 MIT-Indoor

Finally, we perform evaluation on the MIT Indoor scenes dataset. MIT Scenes is a collection of 15620 images of indoor scenes from 67 classes such as bakery, bedroom, church, bar and classroom. The 67 classes can also been seen in 5 superclasses: store, home, public spaces, leisure and working spaces. As with UIUC-sports, the images are resized so that the longest side is no larger than 300 pixels with preserved aspect ratio. For evaluation, we use the pre-defined train/test split given by Quattoni and Torralba [2009], which contains 80 training images from each of the 67 classes and 20 test images from each class.

Figure 4.3 shows a table of the best results obtained on this dataset. Our best performance of 42.6% accuracy outperforms all existing methods except for the deformable parts based models combined with Gist color descriptors as well as the recently proposed hierarchical matching pursuit. We note again that our features were learned on the Mirflickr dataset and had no knowledge of the indoor scene classification task that they were to be used for. Compare this to the deformable parts based model, which would require indoor scenes for training. We hypothesize that further improvements can be made by adding the MIT-scenes training images to the dictionary learning

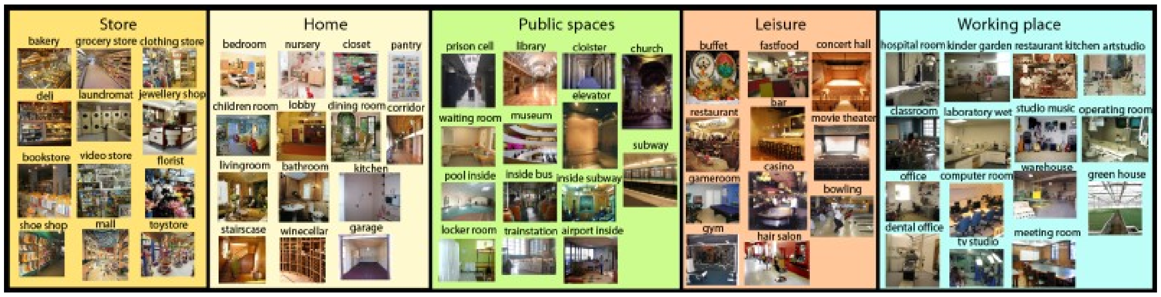


Figure 4.5: The five superclasses of MIT-Indoors used for visualization.

Table 4.3: A selection of the best results obtained on MIT-Indoor

Method	Accuracy
HoG [Pandey and Lazebnik, 2011]	22.8%
MM-scene [Zhu et al., 2010]	28.0%
Gist-color [Pandey and Lazebnik, 2011]	29.7%
Spatial pyramid [Pandey and Lazebnik, 2011]	34.4%
Object bank [Li et al., 2010]	37.6%
DMP + Gist-color + Spatial pyramid [Pandey and Lazebnik, 2011]	43.1%
Hierarchical matching pursuit, 2 modules [Bo et al., 2012a]	47.0%
1st module + SVM	35.4%
1st + 2nd module + t-CMM + k-NN	39.3%
1st + 2nd module + t-MCML + k-NN	42.2%
1st + 2nd module + SVM	42.6%

procedure. Figure 4.6 shows the t-SNE embedding of learned 50 dimensional features using five supercategories: store, home, public spaces, leisure and workplace. Each of the 67 categories is placed into one of the supercategories according to Figure 4.5. Observe that classes from some categories may fall into others. For example, gamerooms are classified as leisure but may also be part of a home.

4.4 Conclusion

In this chapter we introduced a new representation learning algorithm called the deep matching pursuit network, as a multi-module extension to the networks of Coates and Ng [2011a] to larger images. We showed that even on larger images one may obtain highly discriminative, low dimensional representations using linear embeddings. Furthermore, self-taught learning may be successfully applied even in settings where the classification problem is geared towards a specific kind

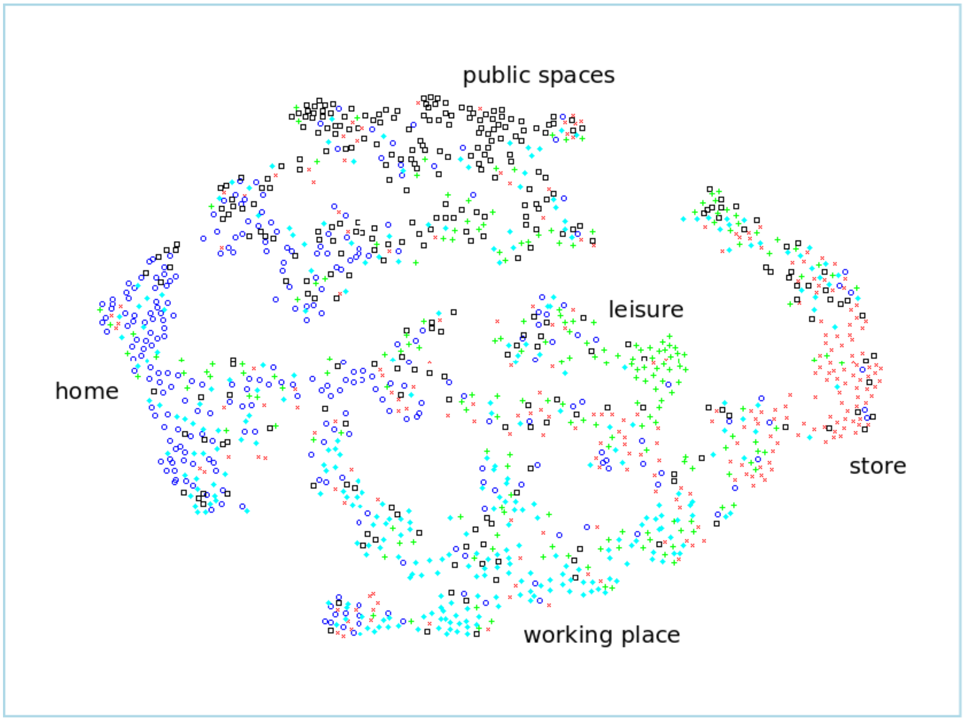


Figure 4.6: t-SNE embedding of the learned 50 dimensional features on the MIT-Scenes test set.

of scene, such as indoor rooms. Finally, we showed that on larger images, deeper networks can be trained without the use of receptive field learning, a result which did not hold on tiny images [Coates and Ng, 2011b].

Our immediate attention for future work is to adapt the same principles used by the hierarchical matching pursuit network to our own. This includes training separate dictionaries for each channel and utilizing multilayer pooling. We also intend to adapt the receptive field learning of Coates and Ng [2011b] in attempts to train a third module. Initial attempts were unsuccessful for the reasons of small spatial dimensions and large number of channels in the second module feature maps.

On datasets like MIT-indoors where the superlabels are not strict, it becomes difficult to learn an appropriate visualization in a two dimensional metric space. This is hampered by the triangle inequality. For example, a buffet and a gameroom are both classified as leisure, while the gameroom could be part of a house. If it was part of a house, it would also be similar to a garage. Yet, the buffet and the garage do not have anything in common. Thus, if an embedding tries to map a buffet close to a gameroom, and a gameroom close to a garage, then by the triangle inequality the buffet will be placed near the garage. This type of scenario was well described in the context of word embeddings [van der Maaten and Hinton, 2011]. In order to deal with this difficulty, van der Maaten and Hinton [2011] proposed a multiple maps version of t-SNE. Multiple maps t-SNE was designed for constructing multiple embeddings of data for cases where the data may not naturally lie in a metric space. Future work would involve constructing multiple map variations of MIT-indoor objects amongst other datasets of similar nature.

Chapter 5

Applications to Auto-Annotation and Retrieval

5.1 Introduction

Image auto-annotation is a multi-label classification task that involves assigning a set of tags from a vocabulary V to an image. Figure 5.1 illustrates this task. Image auto-annotation is challenging for several reasons. First, tag vocabularies are often large and can contain hundreds to thousands of tags. Second, tags may describe both local and global properties of an image. For example, a tag might indicate whether there exists a building (local) or whether the image is outdoors (global). Finally, annotations are often noisy which can include having misspelled tags or multiple tags of the same meaning. Furthermore, tags that are assigned to an image by hand may not be inclusive of all relevant tags from the vocabulary.

Due to this difficulty, it has been common for many of the best performing algorithms [Makadia et al., 2008, Nakayama., 2011, Guillaumin et al., 2009, Tsai et al., 2011, Weston et al., 2010] to fix an often large number of hand-crafted features to describe image characteristics and instead focus on the tagging algorithm itself. Zhang et al. [2010] performed an analysis of feature importance for annotation and observed that the choice of discriptors was dataset dependent. The authors also observed that many features led to redundancy such as the use of HSV and LAB color transforms.

Image auto-annotation has received little attention from the deep learning community. Yet, it is one area of vision that could significantly benefit from having a feature learning algorithm and removing the need for computing over a dozen hand-crafted features and the need for feature selection. In the chapter we attack this problem using the deep matching pursuit network developed in the previous chapter. For performing annotation, we use the TagProp algorithm [Guillaumin

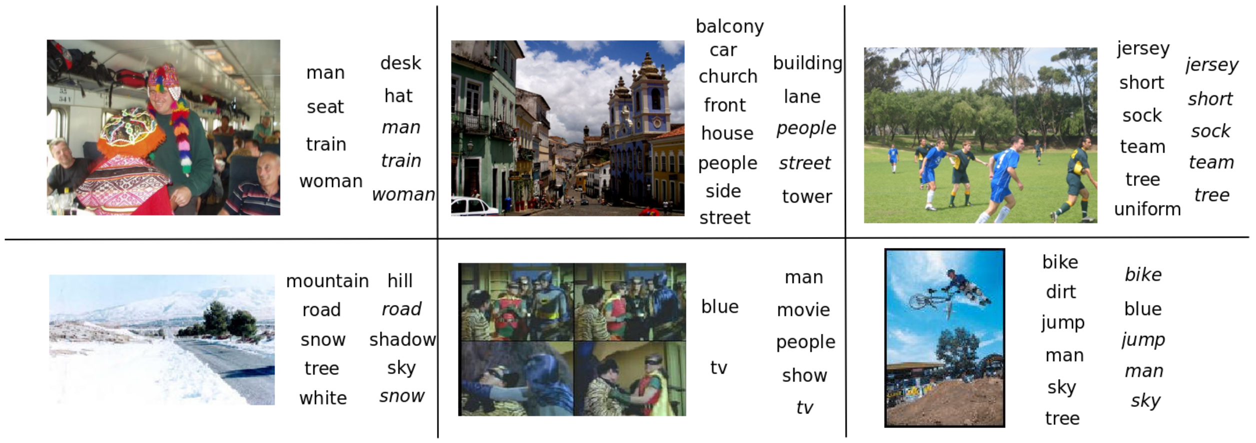

man seat train woman
desk hat *man* *train* *woman*
balcony car church front house people side street
building lane *people* *street* tower
jersey short sock team tree uniform
jersey *short* *sock* *team* *tree*
mountain road snow tree white
hill *road* shadow sky *snow*
blue tv
man movie people show *tv*
bike dirt jump man sky tree
bike blue *jump* *man* *sky*

Figure 5.1: Sample annotation results on IAPRTC-12 (top) and ESP-Game (bottom) using TagProp when each image is represented by a 256-bit code. The left column of tags is the gold standard and the right column are the predicted tags. Predicted tags that are italic are those that are also gold standard.

et al., 2009] that has enjoyed state-of-the-art performance on multiple benchmarks. Specifically, we perform the same set of experiments as was done by Makadia et al. [2008], only replacing the hand-crafted features with our learned representations. We evaluate the performance of the learned features in combination with TagProp on three datasets: Natural Scenes, IAPRTC-12 and ESP-Game. On all datasets we either outperform or compete with existing methods across standard evaluation metrics when over a dozen hand-crafted representations are used for training.

Recently, much emphasis on annotation research has gone towards algorithms that can scale to web size databases containing millions of images. Such algorithms include the visual synsets of Tsai et al. [2011] and the use of joint word-images embeddings of Weston et al. [2010]. Along with using the learned representations from the DMP for annotation, we also consider performing annotation when each image is represented as 256-bit code. Torralba et al. [2008] performed an extensive analysis of image retrieval and reported that linear search with Hamming distance on binary representations could still be computed efficiently even on databases of millions of images. Up until now we have focused mostly on learning supervised, low dimensional embeddings of images. Now we consider learning a sparse unsupervised binary representation. Specifically, we utilize an autoencoder with a single hidden layer to learn binary codes from the high dimensional representations of the DMP. We evaluate the performance of TagProp for annotation when our binary codes are used as input. Since TagProp operates on pairwise distances, an efficient low-level implementation of Hamming distance computation would allow our approach to scale to

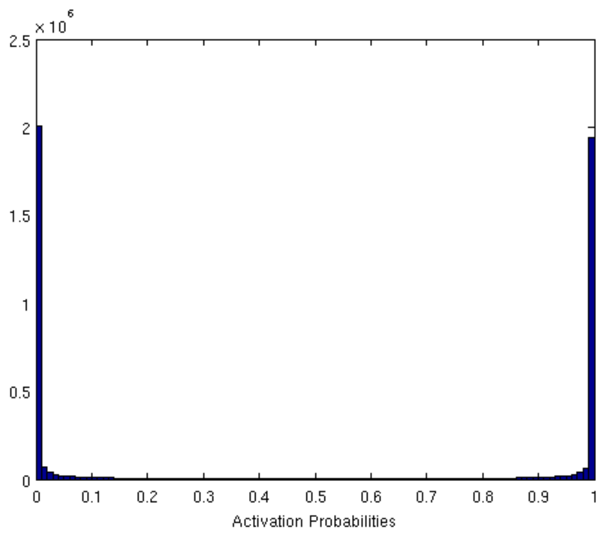


Figure 5.2: Coding layer activation probabilities.

massive datasets. To our surprise, we observe only a small reduction in annotation performance indicating that the binary codes effectively capture high level image semantics. Finally, we perform qualitative analysis of our binary codes for image retrieval in an unsupervised setting. We note that our approach is the first to learn binary codes from full size images without the use of image descriptors. Learning codes from descriptors such as GIST [Oliva, 2005] introduces too strong of a bottleneck too early in the pipeline. In our approach the bottleneck comes after learning multiple modules of representations.

The rest of the chapter proceeds as follows. We first describe in detail the autoencoder used to learn binary codes from the output of the deep matching pursuit network. Next, we review the TagProp algorithm as was introduced by Guillaumin et al. [2009]. This is followed by experimental evaluation and conclusion.

5.2 Learning Binary Representations

Let $f \in \mathbb{R}^{d_m}$ denote the standardized output for an image I of a one or two module architecture of the previous chapter. A binary representation for I is constructed by using f to be input to an autoencoder with a single hidden layer. More specifically, the code b is given by $b = \text{round}(\sigma(f))$ where $\sigma(f^{(i)}) = (1 + \exp(Wf^{(i)} + \beta))^{-1}$, $W \in \mathbb{R}^{d_b \times d_m}, \beta \in \mathbb{R}^{d_b}$ and d_b is the number of bits. Using a linear output layer, our objective is to minimize the mean squared error of the inputs and their reconstructions given by $\frac{1}{m}\sum_i \left[(\tilde{W}\sigma(f^{(i)}) + \tilde{\beta}) - f^{(i)}\right]^2$, where $\tilde{W} \in \mathbb{R}^{d_m \times d_b}, \beta \in \mathbb{R}^{d_m}$ are the second layer weights and biases respectively. The objective is minimized using standard backpropagation.

Ideally, we would perfer the hidden layer activation probabilities to be as close to zero or one as possible. As is, the optimization does not take into consideration the rounding operation, so there is no guarantee that the hidden layer probabilities would follow such a distribution. We follow Salakhutdinov and Hinton [2009] and use "deterministic additive Gaussian noise" in the hidden layer during a forward pass. Salakhutdinov and Hinton [2009] used this operation when training a deep autoencoder in order to obtain binary codes for semantic hashing. We observed that zero mean and unit variance was sufficient to force the activations to be near zero or one. To make the noise "deterministic", we randomly sample from a standard Gaussian before training and use the same noise at each iteration during backpropagation. This is necessary in order for the optimization to perform smoothly. Figure 5.2 displays sample hidden layer activation probabilities after training.

5.3 The TagProp (Tag Propagation) Algorithm

Let I denote a list of images and V a vocabulary of tags. Our goal at test time, given a new image i', is to assign a subset of tags $v \in V$ for which v is most descriptive of image i'. For performing this annotation, we make use of the Tag Propagation (TagProp) algorithm [Guillaumin et al., 2009]. TagProp is a discriminative metric learner that takes as input a set of pairwise distances between training points. Let $y_{iw} \in \{-1, 1\}, i \in I, w \in V$, denote whether tag w is present in image i. Define the probability that tag w is present in image i by

$$\sigma(\alpha_w x_{iw} + \beta_w), \quad x_{iw} = \sum_j \pi_{ij} y_{jw}, \tag{5.1}$$

where $\sigma(z) = (1 + \exp(-z))^{-1}$ is the sigmoid function, (α_w, β_w) are logistic word parameters and $\pi_{ij} = f(d_h)$ are distance based weights for images i and j and some function f. More specifically, π_{ij} is given by

$$\pi_{ij} = \frac{\exp(-d_h(i,j))}{\sum_{j'} \exp(-d_h(i,j'))}, \qquad d_h(i,j) = hd_{ij}, \tag{5.2}$$

where $h \geq 0$ is a scalar parameter on the distance d_{ij} between images i and j, with $h \geq 0$ to enforce that $\pi_{ij} \geq 0$. Let $\theta = (\alpha_w \forall w \in V, \beta_w \forall w \in V, h)$ denote the model parameters which we would like to optimize for. Optimization is done as to maximize a quasi-likelihood on the data given by

$$\mathcal{L} = \sum_{i,w} c_{iw} \log p(y_{iw}), \tag{5.3}$$

where $c_{iw} = \frac{1}{n_+}$ if $y_{iw} = 1$ and $\frac{1}{n_-}$ otherwise with n_+ indicating the total number of annotations for word w and n_- indicating the number of times w was not used for annotation. These are used as a means of weighting words based on their occurrence as to give more weight to rarer tags. Optimization is done using a projected gradient while enforcing the positivity constraint on h.

We denote the distance d_{ij} as the *base* distance between images i and j. As described so far, we assume only a single base distance is given between images. This is easily extended to handle multiple distances by replacing the scalar h with a parameter vector and setting $d_h(i,j)$ to be a weighted combination of base distances. Indeed, it was these multiple distances combined with the logistic word models that led to the best performance when several image descriptors were used by Guillaumin et al. [2009]. In our case, we use the Euclidean distance as our base. When images are represented as binary codes, we utilize the Hamming distance as our base. Distances are only considered between the k nearest neighbors, where k is chosen using e.g. cross validation.

5.4 Experiments

Experimentation is performed on 3 annotation datasets: Natural scenes [Zhou and Zhang, 2007] [1], IAPRTC-12 [Grubinger et al., 2006] [2] and ESP-Game [Von Ahn and Dabbish, 2004] [3]. The deep matching pursuit network is trained using k_1 = 512 first layer bases, k_2 = 1024 second layer bases, receptive field sizes of 6×6 and 2×2 and tile sizes of 16×16 and 6×6. All images are resized such that the maximum length is no more than 300 with preserved aspect ratio. Self-taught learning is employed for all experiments. As was done in evaluation of the DMP in Chapter 4, the Mirflickr dataset is used for dictionary learning. For autoencoder training, optimization is

[1] `http://lamda.nju.edu.cn/data_MIMLimage.ashx`
[2] `http://imageclef.org/photodata`
[3] `http://hunch.net/~learning/ESP-ImageSet.tar.gz`

performed on minibatches of size no larger than 1000 for 10 epochs using Polak-Ribiere conjugate gradients with three linesearches per update.

5.4.1 Natural Scenes

The natural scenes dataset is a collection of 2000 images depicting one or more of the following scenes: desert, forest, sunset, ocean and mountain. We follow standard protocol and perform evaluation by averaging the results of 10-fold cross validation. Evaluation is done using five common metrics for multi-label classification:

- **Hamming Loss:** Computes the difference between true labels and predicted labels using the XOR operation: $HL(x^{(i)}) = \frac{p^{(i)} \otimes y^{(i)}}{N_c}$, where $x^{(i)}$ is a datapoint, $p^{(i)}$ the predicted full label for $x^{(i)}$, $y^{(i)}$ the target full label and N_c the total number of labels.

- **One Error:** Evaluates whether the most probable label prediction is part of the gold standard: $OE = \frac{1}{N_x} \sum_i \mathbf{1}(l_1(x^{(i)}) \notin y^{(i)})$, where N_x is the number of test set images, l_1 is the first ranked label and $\mathbf{1}()$ is the indicator function.

- **Coverage:** Evaluates how far away the predicted labels are from 'covering' the true labels: $C = \frac{1}{N_x} \sum_i \max_{l \in y^{(l)}}$ rank$(l, x^{(i)}) - 1$, where rank$(l, x^{(i)})$ denotes the position of label l in the ranked list for image $x^{(i)}$.

- **Ranking Loss:** The fraction of labeled pairs between true and irrelevant labels that are in reverse order: $RL = \frac{1}{N_x} \sum_i \frac{F(x^{(i)})}{|y^{(i)}||\bar{y^{(i)}}|}$, where $F(x^{(i)}) = |\{(l, l') :$ rank$(l, x^{(i)}) >$ rank$(l', x^{(i)}), l \in y^{(i)}, l' \in \bar{y^{(i)}}\}|$ is the number of reverse ordered pairs of labels from the ranked list of $x^{(i)}$.

- **Average Precision:** $AP = \frac{1}{N_x} \sum_i \frac{1}{y^{(i)}} \sum_{l \in y^{(i)}} \frac{G(x^{(i)})}{\text{rank}(l, x^{(i)})}$ where $G(x^{(i)}) = |\{l' \in y^{(i)} :$ rank$(l', x^{(i)}) \leq$ rank$(l, x^{(i)})\}|$ is the number of gold standard labels ranked above label l.

The number of nearest neighbors k for training TagProp is chosen through 5-fold cross validation on each of the training folds in order to minimize Hamming loss. A tag is assigned to an image if the probability of a tag given the image exceeds 0.95. Table 6.1 describes the results of our approach to the state-of-the-art on this dataset. The most successful method to date is based on image to class distance learning. The combination of feature learning and annotation with TagProp outperforms all existing approaches on this dataset. We attribute the successful performance

Table 5.1: A selection of the best results obtained on the Natural Scenes dataset. Arrows indicate direction of improved performance.

Method	HL $\downarrow$	OE $\downarrow$	C $\downarrow$	RL $\downarrow$	AP $\uparrow$
ML-KNN [Zhang and Zhou, 2007]	0.169	0.300	0.939	0.168	0.803
ML-I2C [Z. Wang and Chia., 2010]	0.159	0.311	0.883	0.156	0.804
InsDif [Zhang and Zhou., 2007]	0.152	0.259	0.834	0.140	0.830
ML-LI2C [Z. Wang and Chia., 2010]	0.129	0.190	0.624	0.091	0.881
1st Module	0.113	0.170	0.580	0.080	0.895
1st Module, 256-bit	0.113	0.169	0.585	0.082	0.894
1st + 2nd Module	0.100	0.140	0.554	0.074	0.910
1st + 2nd Module, 256-bit	0.106	0.155	0.558	0.075	0.903

of the binary codes to the fact that they were constructed such that Hamming distance would be appropriate. As was the case for evaluation of DMP in Chapter 4, using a second module leads to improved performance.

5.4.2 IAPRTC-12 and ESP-Game

IAPRTC-12 is a collection of 20000 images with a vocabulary size of $|V| = 291$ and an average of 5.7 tags per image. ESP-Game is a collection of 60000 images with $|V|$ = 268 and an average of 4.7 tags per class. Following [Guillaumin et al., 2009] we apply experiments to a subset of 20000 images. Performance is evaluated using four measures: Precision ($\mathbf{P}$), Recall ($\mathbf{R}$), F-measure ($\mathbf{F}$) and the number of recalled tags ($\mathbf{N}+$). Precision, recall and F-measure are micro-averaged. The number of recalled tags indicated the number of tags that occured in training that were recalled at least once on the test set. Following standard procedure, we assign the 5 most probable tags to each image. 5-fold cross validation is used for selecting k for TagProp that maximizes F-measure.

Figures 5.3, 5.4 and 5.5 displays the results of our approach. On both datasets, our results perform comparably to CCD, although is not able to match the recall and N+ values obtained using TagProp and 15 hand crafted features. More importantly, our representations outperform GS, indicating that representation learning for annotation is a considerable alternative as opposed to feature selection of several engineered features. Of interest are also the results obtained when an image is represented with a 256-bit code. While on IAPRTC-12 using binary codes lead to a reduction in performance, on ESP-Game 256-bit codes perform equivalently with our 2 module, high dimensional representations, while obtaining a larger N+ value. Recall that our representations

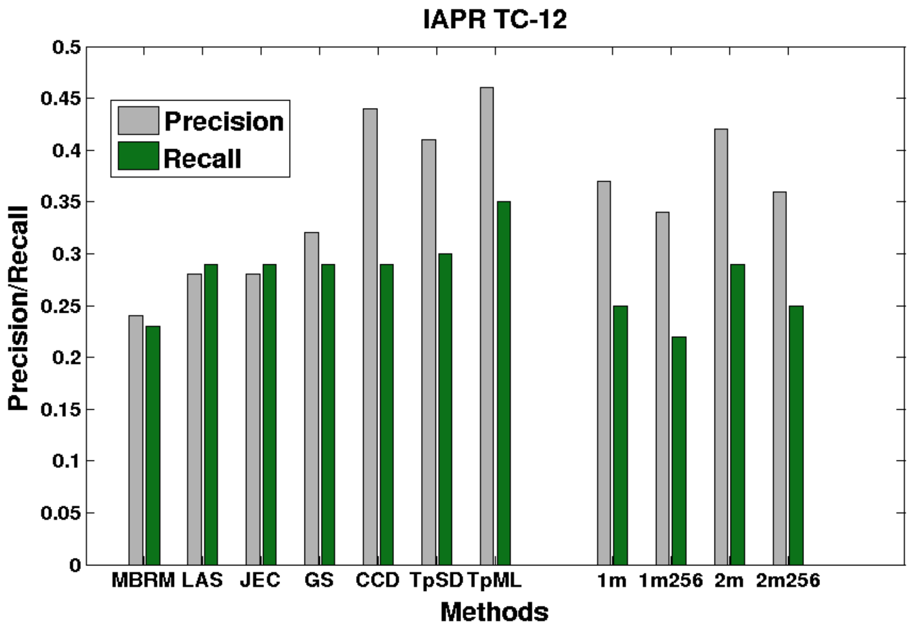


Figure 5.3: Precision and recall values on IAPR TC-12. Methods compared against are MBRM [Feng et al., 2004], LASSO [Makadia et al., 2008], JEC [Makadia et al., 2008], GS [Zhang et al., 2010], CCD [Nakayama., 2011] and TagProp [Guillaumin et al., 2009]. 1m, 1m256 refer to our method with one module and one-module 256-bit codes. Similarly for 2m, 2m256 with 2 modules.

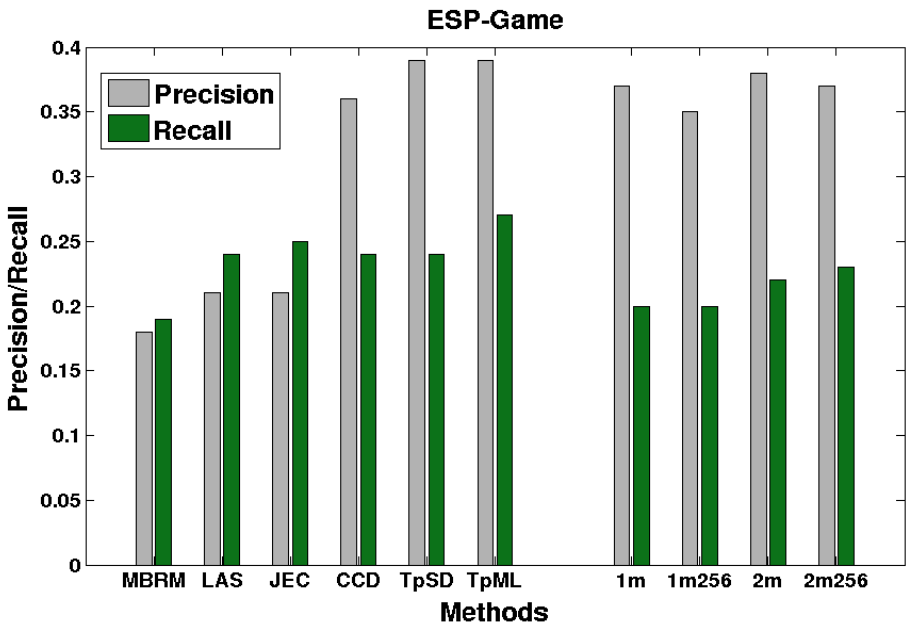


Figure 5.4: Precision and recall values on ESP-Game. Methods compared against are MBRM [Feng et al., 2004], LASSO [Makadia et al., 2008], JEC [Makadia et al., 2008], GS [Zhang et al., 2010], CCD [Nakayama., 2011] and TagProp [Guillaumin et al., 2009]. 1m, 1m256 refer to our method with one module and one-module 256-bit codes. Similarly for 2m, 2m256 with 2 modules.

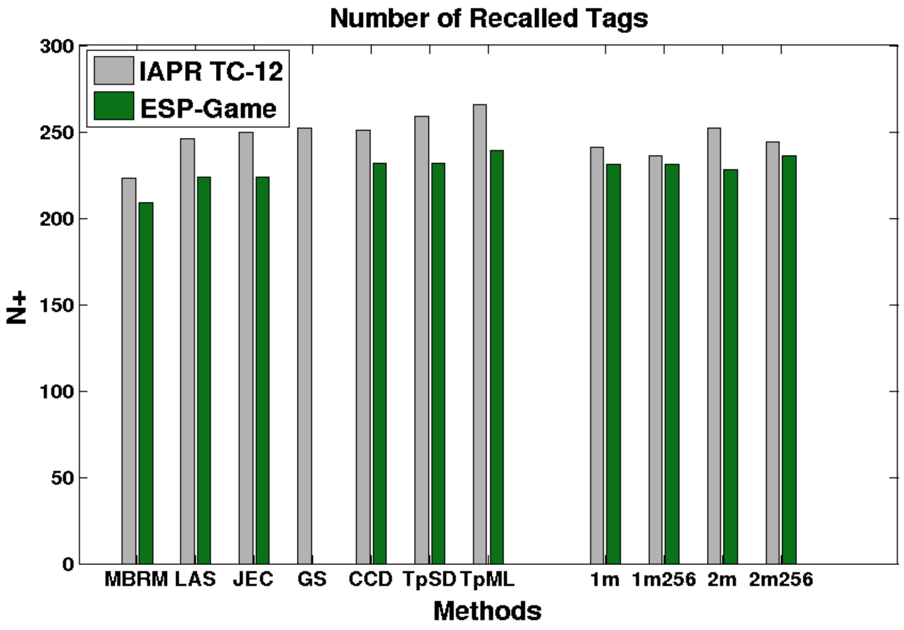


Figure 5.5: The number of recalled tags on IAPR TC-12 and ESP-Game. Methods compared against are MBRM [Feng et al., 2004], LASSO [Makadia et al., 2008], JEC [Makadia et al., 2008], GS [Zhang et al., 2010], CCD [Nakayama., 2011] and TagProp [Guillaumin et al., 2009]. 1m, 1m256 refer to our method with one module and one-module 256-bit codes. Similarly for 2m, 2m256 with 2 modules.

were learned through self-taught learning on Mirflickr, indicating that the algorithm has learned a general enough filter bank for a variety of vocabulary words and images.

5.4.3 Retrieval

Finally, we consider using our learned binary codes for unsupervised retrieval on the IAPRTC-12 and ESP-Game datasets. Figure 5.6 shows sample retrieval results when a query image from the test set is used to retrieve its 4 nearest neighbors from the training set. Qualitatively, we observe that our codes perform well at encoding high level semantic concepts, those that could not be captured through pixel distance.

5.5 Conclusion

In this chapter we showed that features learned using the deep matching pursuit network can be successfully employed on image auto-annotation tasks. In particular, we showed that learned features when used with TagProp can compete with or outperform existing approaches including those

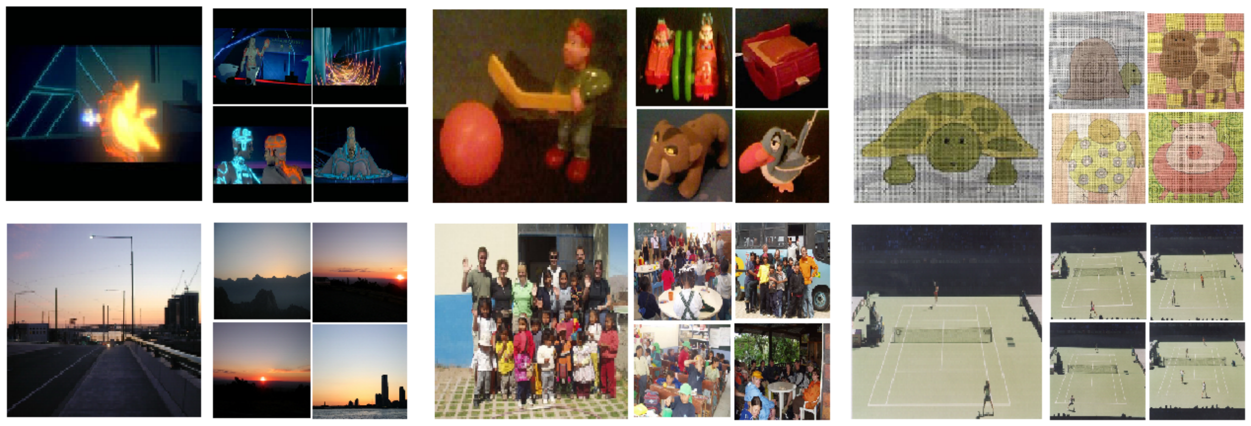


Figure 5.6: Sample (unsupervised) retrieval results when each image is represented using a 256-bit code. The query image is from the test set and used to retrieve the 4 nearest training images based on their Hamming distances.

that use over a dozen hand-crafted features. An important direction for future work would involve scaling our annotation and retrieval system to a more realistic, large-scale setting with millions of images. Moreover, semantic hashing may be employed to perform retrieval in time independent of the size of the database. Such a result would be the first large-scale system for retrieval that did not make use of any hand-crafted representations. In the following chapter, we discuss generalizing the deep matching pursuit networks of Chapter 4 to other modalities such as RGB-D or audio. Future research could involve performing the same set of annotation and retrieval experiments on RGB-D and audio data with applications to robotics, music auto-tagging and music retrieval. It would also be of interest to determine whether further increasing dictionary sizes can lead to a further increase in performance.

Chapter 6

Applications to Multimodal Medical Image Segmentation

6.1 Introduction

The choice of image representation plays a crucial role in the success of medical image segmentation algorithms. Most existing methods utilize hand-crafted features incorporated into an energy-based segmentation method or into a machine learning classifier. Commonly, energy-based methods utilize engineered features such as Gabor filters for texture-based segmentation [Paragios and Deriche, 2002], while machine learning approaches use many more simple features like Haar or steerable filters leaving the classification method to disambiguate the ones that are significant for the segmentation task. Popular examples of machine learning methods are ones based on decision trees [Zheng et al., 2008], random forests [Criminisi et al., 2013] as well as SVMs and conditional random fields [Lee et al., 2005]. Some methods use very specialized filters designed for a particular task, such as extracting linear structures based on eigenvalues of the image Hessian matrix [Frangi et al., 1998].

In this chapter we introduce the multi-scale deep matching persuit network that utilizes features learned from multiple scales and depths. The key features that make our method fast and thus suitable for medical data are detailed below and can be summarized as: (1) patch-based, (2) stage-based system and a (3) fast dictionary learning method. Table 6.1 summarizes and distinguishes four types of feature learning architectures for medical images. Convolutional sparse coding algorithms, such as those used by Rigamonti and Lepetit [2012] and Rigamonti et al. [2011], differ from standard sparse coding methods as convolution is incorporated into the optimization procedure. The third architecture describes convolutional networks, used by Ciresan et al. [2012] for

Table 6.1: A comparison of different feature learning architectures that have been applied to medical image segmentation: Y is yes, N is no and S is sometimes. Multi-scale and multi-depth methods can often improve performance while patch-based and stagewise learning improve speed. Here sparse coding refers to any method that aims to learn a filter bank with a sparsity constraint.

Method	Patch-based	Multi-scale	Multi-depth	Stagewise
Sparse coding	Y	N	N	Y
Convolutional sparse coding	N	N	N	Y
Convolutional networks	N	S	Y	N
Proposed approach	Y	Y	Y	Y

electron microscopy image segmentation, which are learned jointly with supervision. While convolutional networks are often very effective, jointly training the whole model can be time consuming. Furthermore, convolutional networks require many labeled examples in order to avoid overfitting. The last architectures illustrates our proposed framework. Features are learned one stage at at time using patch-based learning at multiple scales. Since the model does not require joint learning, features can be learned efficiently and quickly. Our framework is largely influenced by the successful applications of patched-based filter learning of Coates and Ng [2011c] for object recognition, which utilizes k-means and gain-shape vector quantization combined with convolutional extraction. The emphasis of this work being the importance of the feature encoding as opposed to the filter learning algorithm itself. Due to this, we suggest that more expensive convolutional filter learning is unnecessary, so long as a proper encoding is performed after learning.

To test the performance of our proposed framework, we applied our method to two very different medical datasets of the MICCAI grand challenges: vessel segmentation of the lung (VESSEL12) and multimodal brain tumor segmentation. Our method achieves competitive performance on both tasks, compared to the existing leaderboard approaches for these challenges. Furthermore, our system is able to learn features in under ten minutes on both challenges.

6.2 Method

We assume we are given m volumes with s modalities $\{\{V_i^{(j)}\}_{j=1}^{s}\}_{i=1}^{m}$ with each $V_i^{(j)} \in \mathbb{R}^{n_V \times n_H \times n}$ where $n_V \times n_H$ is the spatial dimension of a slice and n is the number of slices. For simplicity, we assume each volume $V_i^{(j)}$ has dimensionality $n_V \times n_H \times n$ although this need not be the case. As a specific example, brain tumor segmentation tasks can use $s = 4$ modalites consisting of

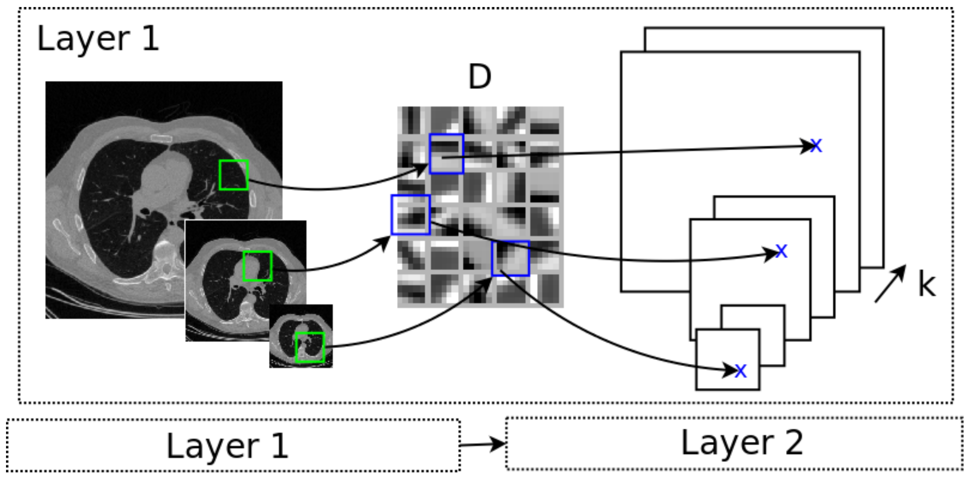


Figure 6.1: Visualization of our feature learning approach. Each volume slice is scaled using a Gaussian pyramid. Patches are extracted at each scale to learn a dictionary D using OMP. Convolution is performed over all scales with the dictionary filters, resulting in Γk feature maps. After training the first layer, the feature maps can then be used as input to a second layer.

FLAIR, T1, T2 and post-Gadolinium T1. The general outline of our feature learning framework is as follows:

- Extract multimodal patches at multiple scales using a Gaussian pyramid.
- Learn a filter bank using orthogonal matching pursuit.
- Convolutionally extract feature maps using the learned filters as kernels.
- Repeat the above steps, using the computed features maps as input to the next layer. The number of feature maps (next layer modalities) corresponds to the number of filters.

Each of the following subsection describes one of the above operations in detail.

6.2.1 Pre-processing and dictionary learning

Given a volume V, a Gaussian pyramid with Γ scales is applied to each modality of each slice. Let $\{p^{(1)}, \ldots, p^{(m_P)}\}$ denote a set of m_P patches randomly extracted from the scaled volumes. Each patch $p^{(l)}$ is of spatial dimension $r \times c \times s$ where $r \times c$ is the receptive field size and s is the number of modalities. These patches are then flattened into column vectors. Per patch contrast normalization and patch-wise mean subtraction is performed. For dictionary learning we use orthogonal matching pursuit (OMP) as was done in Chapter 3 to obtain a solution to the following optimization problem:

$$\begin{aligned} \underset{D, x^{(i)}}{\text{minimize}} \quad & \sum_{i=1}^{m_P} ||Dx^{(i)} - p^{(i)}||_2^2, \\ \text{subject to} \quad & ||D^{(l)}||_2^2 = 1, \forall l, \\ & ||x^{(i)}||_0 \leq q, \forall i, \end{aligned} \tag{6.1}$$

where $D \in \mathbb{R}^{n_P \times k}$ and $D^{(l)}$ is the l-th column of D.

6.2.2 Convolutional feature extraction

Let T_j^γ denote a volume slice of modality j and scale γ. Each $r \times c \times s$ patch in T_j^γ is pre-processed by contrast normalization and mean subtraction. Let $D_j^{(l)} \in \mathbb{R}^{r \times c}$ denote the l-th basis for modality j of D. We will define the feature encoding for basis l to be given by:

$$f_l^\gamma = \sum_{j=1}^{s} T_j^\gamma * D_j^{(l)}, \tag{6.2}$$

where * denotes convolution. The resulting feature maps $\{f_l^\gamma\}_{l=1}^k$ are of the same spatial dimensions as T_j^γ. The feature maps are finally upsampled to the original $n_V \times n_H$ spatial dimension. Figure 6.1 illustrates our approach.

6.2.3 Stacking multiple layers

Our described setup for feature learning has involved scaling, dictionary learning and convolutional extraction. Just as the volumes slices were inputs to a first layer with s modalities, the upsampled output feature maps $\{\{f_l^\gamma\}_{\gamma=1}^\Gamma\}_{l=1}^k$ may be seen as inputs to a second layer but with Γk modalities. All the same described operations are applied a second time resulting in additional second layer output feature maps. These groups of feature maps can be concatenated together resulting in a total number of $\Gamma_1 k_1 + \Gamma_2 k_2$ feature maps, where Γ_1, k_1 are the number of first layer scales and filters while Γ_2, k_2 are the number of second layer scales and filters. Thus each pixel in a volume slice can be represented as a $\Gamma_1 k_1 + \Gamma_2 k_2$ dimensional feature vector.

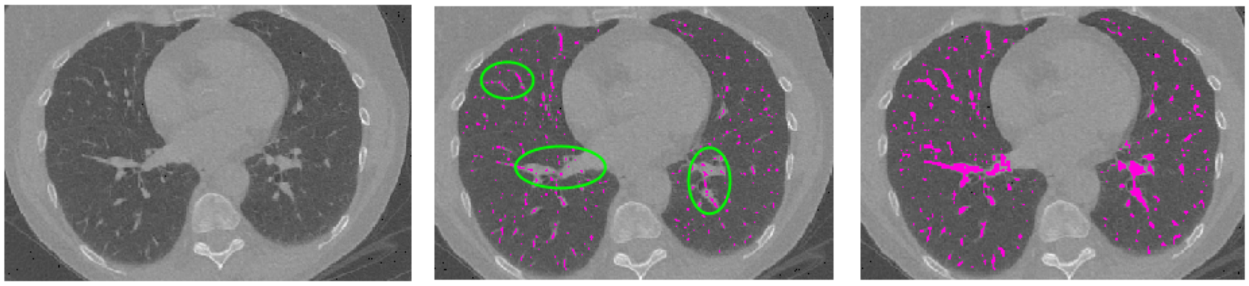


Figure 6.2: Visualizing the importance of scale and depth for vessel segmentation.

6.3 Experiments

We perform experimental evaluation using data from two MICCAI grand challenges: vessel segmentation of the lung [1] and multimodal brain tumor segmentation [2].

6.3.1 Vessel segmentation

The vessel segmentation challenge consists of 20 volumes of CT scans to segment with 3 additional volumes that include 882 labeled pixels based on the agreement of at least 3 experts. Each slice is of size 512×512 with each volume containing a few hundred slices. We performed feature learning with 2 depths, 6 scales, a receptive field size of 5×5, 32 first layer filters and 64 second layer filters. The final feature vector is thus of size $6 \times (32 + 64) = 576$. In order to perform segmentation, we extracted features for the existing labeled pixels and trained a L2-regularized logistic regression classifier, using 10-fold cross validation in order to tune the L2 hyperparameter. Each pixel of a new slice is then classified, resulting in a probability of whether or not the pixel is a vessel. For our submission to the challenge, the probabilities are scaled and rounded to unsigned 8-bit integers as requested.

Figure 6.2 illustrates the importance of adding depth and segmentation. The first image is the CT scan to segment vessels. The second image shows segmentation when neither depth or scale is added while the third image shows segmentation with added depth and scale. Without scale, larger vessels are less likely to be segmented while without depth, segmentation is much more scattered and less contiguous. For visualization purposes, a pixel is labeled as being a vessel if the probability of a vessel given the pixel features is greater than 0.5.

[1]`http://vessel12.grand-challenge.org/`

[2]`http://www2.imm.dtu.dk/projects/BRATS2012/`

Table 6.2: The top 5 results from the VESSEL12 challenge leaderboard. Anon_feat_learning refers to our results.

Team	Method type	score
Anon_feat_learning	feature learning + classification	**0.986**
LKEBChina	Krissian-inspired vesselness	0.984
FME_LungVessels	Frangi vesselness + region growing	0.984
LKEBChina	Krissian-inspired vesselness with bi-Gaussian kernel	0.981
FME_LungVessels	Frangi vesselness + region growing (raw)	0.981

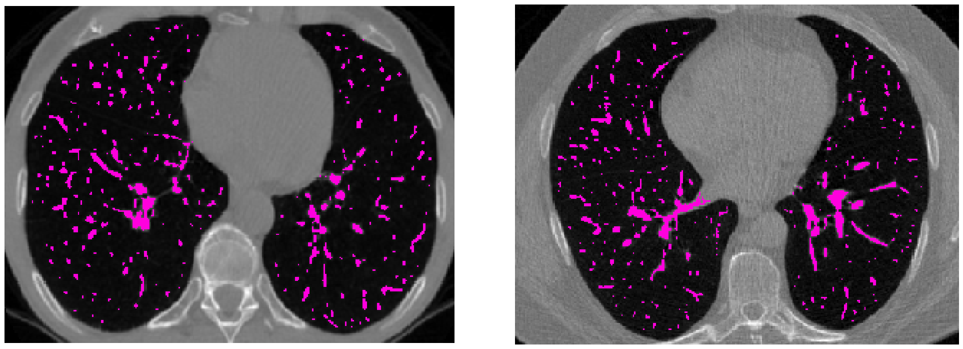


Figure 6.3: Sample vessel segmentation results on two held-out patients.

Table 6.2 shows the top 5 performing methods on the VESSEL12 challenge. Our proposed method tops all existing approaches. The top performing methods in the competition are largely based on the use of Frangi [Frangi et al., 1998] and Krissian vesselness [Krissian et al., 2000] all of which derive structural properties from the eigenvalues of the Hessian.

6.3.2 Brain tumor segmentation

Our second evaluation is performed on the BRATS2012 multimodal brain tumor segmentation challenge. Due to BRATS2012 site maintenance, test volume labels were unavailable. Instead we perform evaluation using leave-one-out cross validation on the training set. Two types of volumes are evaluated: high-grade and low-grade. Each volume voxel is labeled as being one of three classes: tumor, edema and other. We utilized our approach with one scale and two depths, with 16 bases in each depth for a total of 32 features. A 2-hidden layer network with dropout [Hinton et al., 2012] is used to make predictions. Within each training fold, 10-fold cross validation is used to select the dropout parameters.

Table 6.3 shows our results in comparison to the top 2 methods in the competition. We note

Table 6.3: Comparison against the top two performers in the BRATS2012 competition. HG and LG stand for high-grade and low-grade, respectively.

Team	region	mean dice coeff.	region	mean dice coeff.
Anon_feat_learning	HG edema	0.485	LG edema	0.250
Bauer et al.	HG edema	0.536	LG edema	0.179
Zikic et al.	HG edema	**0.598**	LG edema	**0.324**
Anon_feat_learning	HG tumor	0.470	LG tumor	**0.406**
Bauer et al.	HG tumor	**0.512**	LG tumor	0.332
Zikic et al.	HG tumor	0.476	LG tumor	0.339
Anon_feat_learning	HG GTV	0.720	LG GTV	0.494

again that our comparison is not on the same held-out data. None-the-less, our results are competitive with the top performing methods.

6.4 Conclusion

In this paper we proposed a domain independent approach to segmenting medical images. Our approaches involves learning feature representation from both multiple scales and depth that are compatible with existing classification and energy-based segmentation methods. We obtain the best performing result on the VESSEL12 challenge and obtain competitive results on the BRATS2012 multimodal brain tumor segmentation challenge. For future work we intend to further evaluate our approach on additional challenge problems. We also intend to study various transfer learning scenarios between domains and modalities.

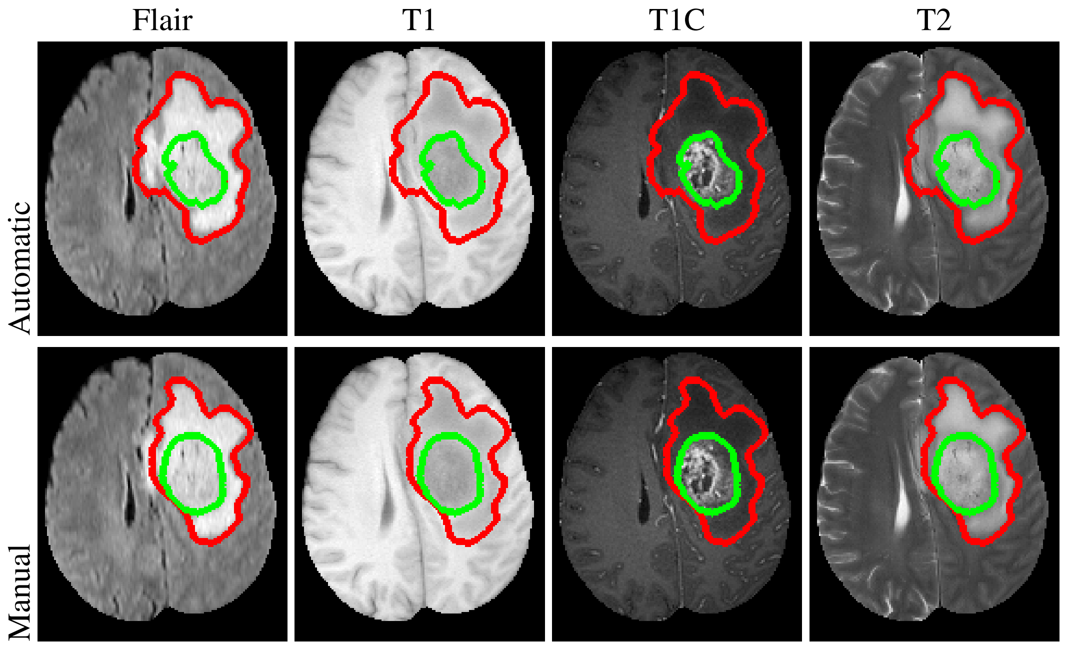


Figure 6.4: Sample brain tumor segmentation result from a held-out test patient. The top row is the results of our approach with the bottom row corresponding to an expert segmentation. The region outlined in yellow is the tumor and the edema is outlined in red.

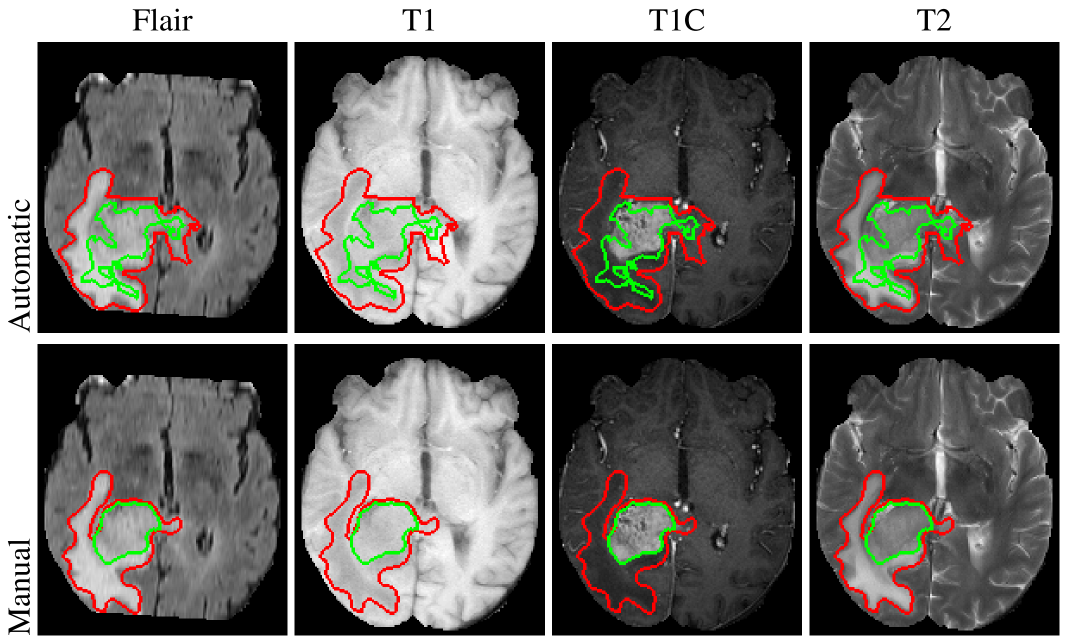


Figure 6.5: Sample brain tumor segmentation result from a held-out test patient. The top row is the results of our approach with the bottom row corresponding to an expert segmentation. The region outlined in yellow is the tumor and the edema is outlined in red.

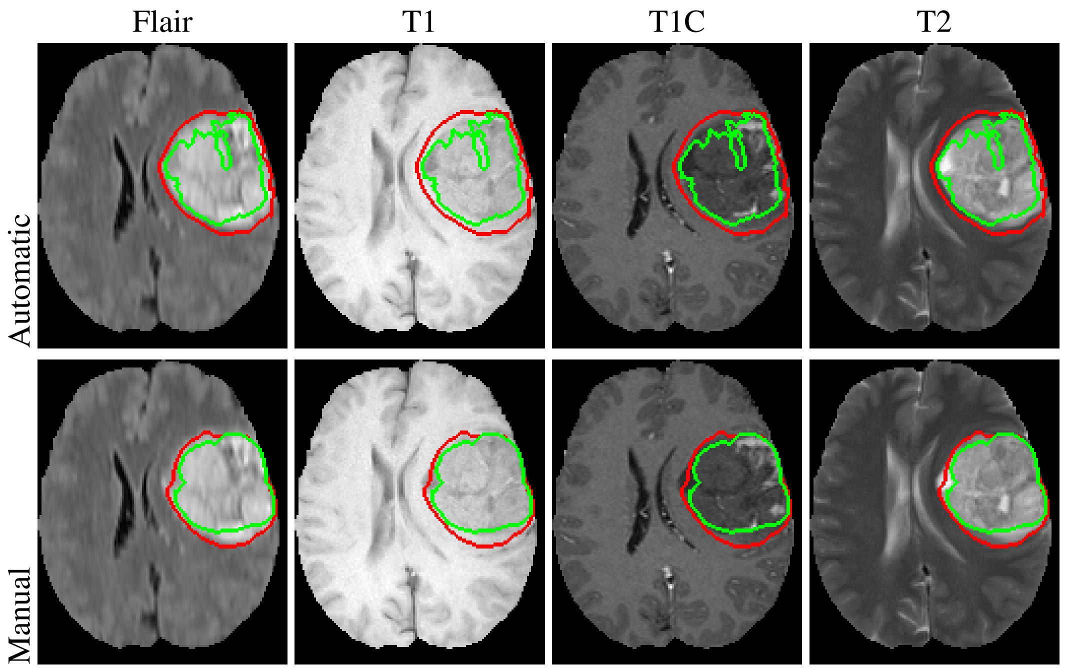


Figure 6.6: Sample brain tumor segmentation result from a held-out test patient. The top row is the results of our approach with the bottom row corresponding to an expert segmentation. The region outlined in yellow is the tumor and the edema is outlined in red.

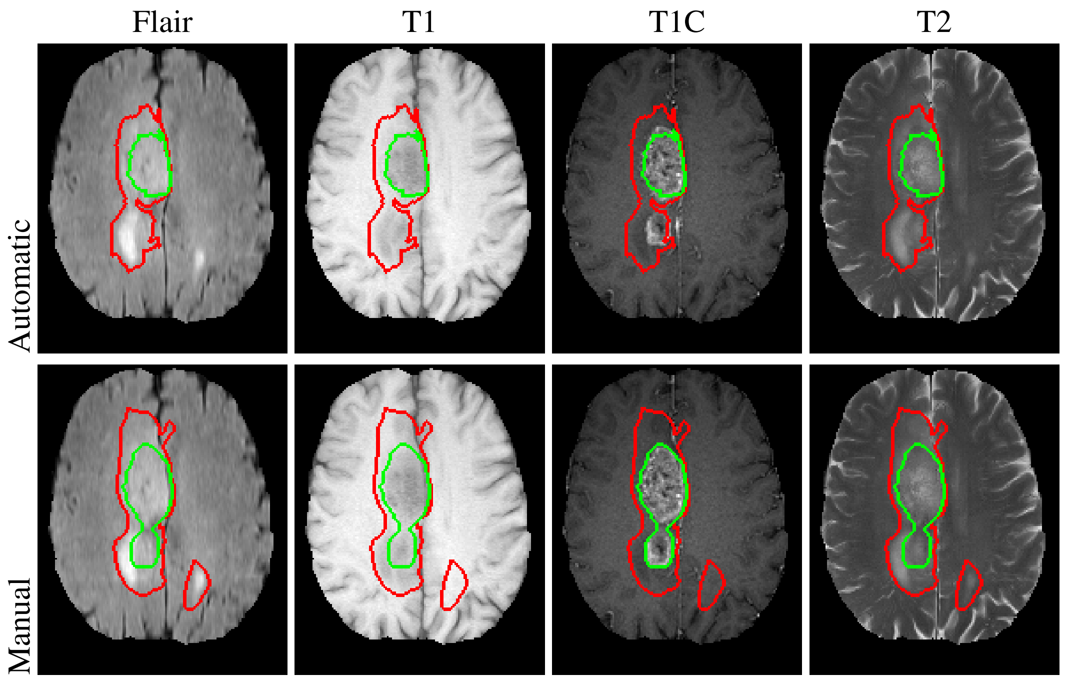


Figure 6.7: Sample brain tumor segmentation result from a held-out test patient. The top row is the results of our approach with the bottom row corresponding to an expert segmentation. The region outlined in yellow is the tumor and the edema is outlined in red.

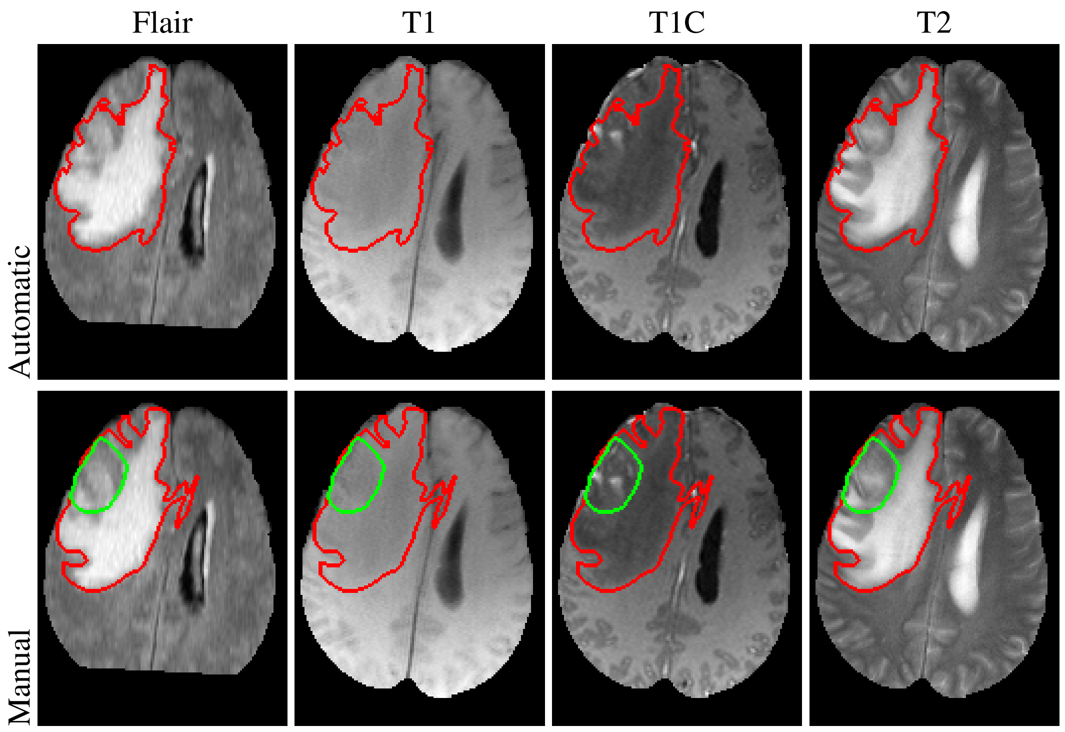


Figure 6.8: Sample brain tumor segmentation result from a held-out test patient. The top row is the results of our approach with the bottom row corresponding to an expert segmentation. The region outlined in yellow is the tumor and the edema is outlined in red.

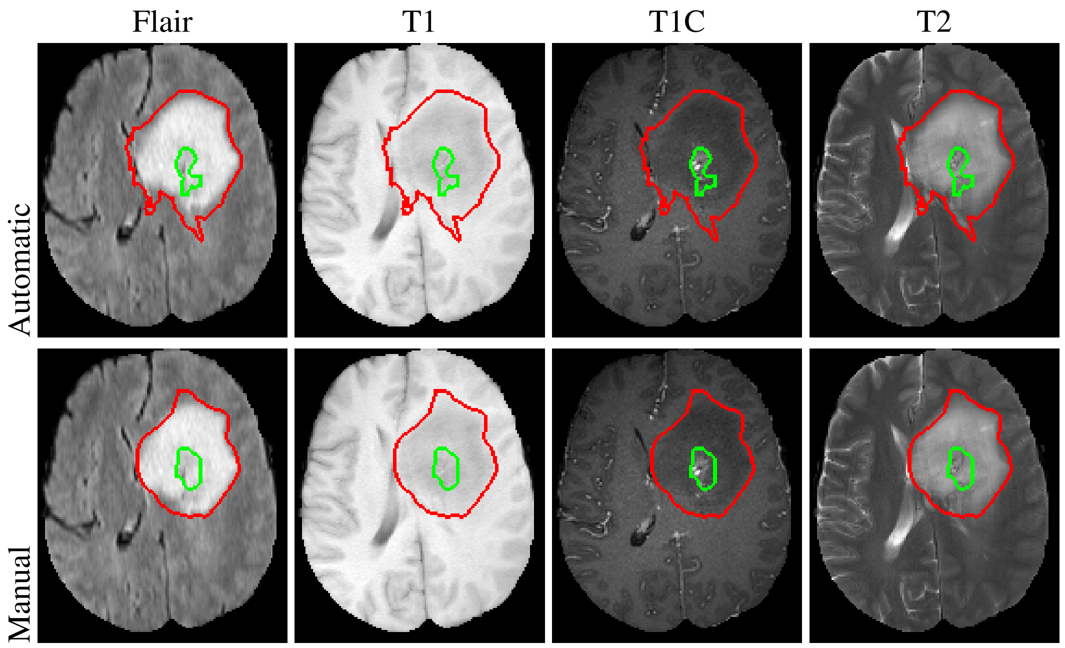


Figure 6.9: Sample brain tumor segmentation result from a held-out test patient. The top row is the results of our approach with the bottom row corresponding to an expert segmentation. The region outlined in yellow is the tumor and the edema is outlined in red.

Chapter 7

Conclusion

In this dissertation we proposed algorithms for learning two kinds of feature representations from images: semi-supervised dense low dimensional embeddings and unsupervised sparse binary codes. We then proposed a multi-scale variation of the DMP network for segmenting medical images. These features were learned by combining a proposed two-module network with either a linear Mahalanobis metric learner or an autoencoder, learned directly from the pixel level without the use of hand-crafted features.

7.1 Future Work

One key area of future work is adapting the DMP architecture to alternative modalities not covered in this thesis. Since these representations are essentially learned from scratch, it is sensible that the same architectures and pre-processing mechanisms can be used on other modalities. Here we briefly describe how the deep matching persuit network can be modified to handle three additional modalities: RGB-D images and audio.

7.1.1 RGB-D Images

RGB-D cameras and images consist of RGB channels as well as an additional fourth channel, containing depth information. Figure 7.1 illustrates sample images as well as their depth maps [1]. Research using RGB-D imaging has gained immense popularity with the release of the Microsoft Kinect and is having impacts on pose recognition [Shotton et al., 2011], object recognition [Bo et al., 2012a], modelling indoor environments [Henry et al., 2010] and grounded attribute learning

[1]Images obtained from `http://www.cs.washington.edu/rgbd-dataset/`

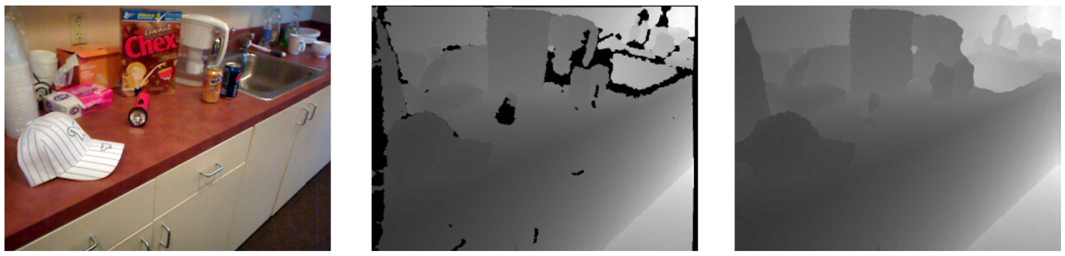


Figure 7.1: A sample image, its depth map and a smoothed depth map. Black pixels in the second images denote unknown depth.

[Matuszek et al., 2012], among others. A natural question to consider is whether or not feature learning algorithms designed for RGB images can be easily adapted for additional depth information. Lai et al. [2011] introduced a new RGB-D dataset consisting of various households objects placed on a rotating platform. The dataset also contains annotated videos of an RGB-D camera containing the featured objects.

For performing feature learning on RGB-D images, Blum et al. [2012] introduced the convolutional k-means descriptor and performed a set of experiments illustrating that feature learning with depth information can lead to improved performances in recognition, simply treating the depth map as an additional fourth channel. Alternatively, Bo et al. [2012a] adapted their hierarchical matching pursuit networks to RGB-D images by training four separate dictionaries: grayscale, RGB, depth and surface normals. The features extracted from all dictionaries were concatenated and used for classification.

7.1.2 Audio and Spectrograms

One of the biggest successes of deep learning algorithms have come from audio and speech. Mohamed et al. [2009] utilized deep belief networks for phone recognition and later constructed the DNN-HMM [Seide et al., 2011] and convex training modules [Deng and Yu, 2011]. Lee et al. [2009b] utilized feature learning from spectrograms using a convolutional deep belief network while Hamel and Eck [2010] applied deep belief networks to music genre classification. Figure 7.2 demonstrates bases learned using sparse coding from spectrograms that roughly correspond to phonemes [2]. To apply the deep matching pursuit network to audio, a spectrogram is computed across overlapping windows, often of a few milliseconds, using a discrete Fourier transform. The

[2]Image obtained from `https://www.ipam.ucla.edu/publications/gss2012/gss2012_10595.pdf`

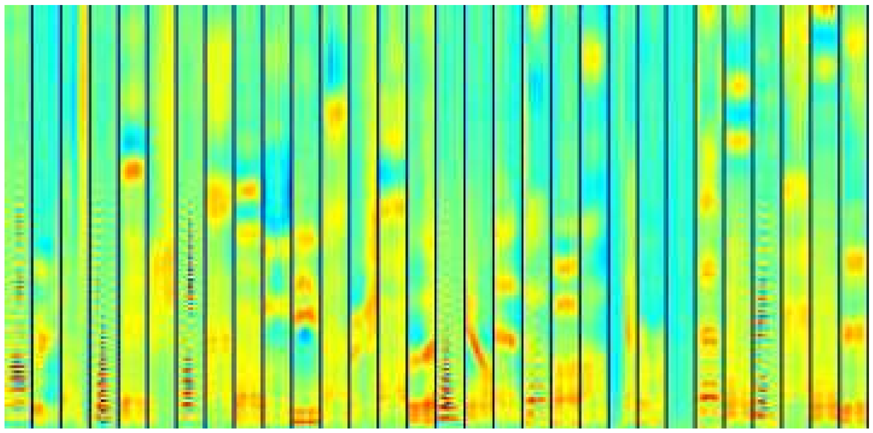


Figure 7.2: Sample bases learned using sparse coding on spectrograms.

result is a one dimensional representation across time where channels correspond to frequencies. Thus, the DMP remains identical to that presented with the exception that the two dimensional spatial convolutions are replaced with one dimensional convolutions across time. Pooling is performed across time for feature aggregation and constructing inputs for training a second module.

One technical detail that remains is in choosing the appropriate regularizer for whitening. Recall that whitening regularization on images had the effect of a low-pass filter [3]. Using a visualization of the eigenspectrum produced from the eigenvalues of the convariance matrix, a regularization may be chosen as to remove as much of the 'long tail' as possible.

7.1.3 Final Thoughts

There are several avenues of future work from which to proceed. The first is to consider developing experiments for large scale tasks such as web-scale annotation. For such methods to be feasible, it would be beneficial to develop implementations of the deep matching pursuit framework to make use of GPUs in order to speed up computation. The biggest bottleneck in computation comes from the number of convolutions that are required in computing feature maps. Such operations can benefit from a substantial speedup on GPUs, allowing us to apply our methods to millions of images in a short time. In particular, given learned basis for one and two module architectures as well as a trained autoencoder to learn binary codes, each image can be processed in an online fashion for which their binary representations can be stored. Only saving the binary features easily

[3]A detailed procedure for selecting an appropriate regularization is described at `http://ufldl.stanford.edu/wiki/index.php/Data_Preprocessing`

allows us to learn representations from massive datasets. Such representations can then be used for annotation and retrieval tasks at a web-scale. Since our architecture may also be used on audio, the same tasks can also be performed for doing music retrieval and annotation.

In terms of medical applications, future work would also involve adapting the multiscale DMP network to handle longitudinal data. For example, if one is given MRI scans of the brain over different times, the growth of a tumor can be more appropriately modeled and perhaps used to try to predict its behavior at future times. The DMP network can also be extended to extract features in 3D instead of 2D, either to model a 3D spatial region or to combine 2D features over different time steps. Such operations are easily generalizable in the DMP framework, as we make no particular assumptions on the dimensionality of the patches.

Finally, we hope to apply the DMP framework to additional recognition tasks or tasks based on weak supervision. One such example could be to perform fine-grained image segmentation using labels from a bounding box. Alternatively, we could integrate our methods with structure segmenters for performing tasks such as extracting writing and text from images. Of great interest would also be to develop a framework that would not only use bottom-up information for inference but also top-down information.

Bibliography

H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Computer Vision–ECCV 2006*, pages 404–417, 2006.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

S. Belongie, J. Malik, and J. Puzicha. Shape context: A new descriptor for shape matching and object recognition. *Advances in neural information processing systems*, pages 831–837, 2001.

Y. Bengio, J.F. Paiement, P. Vincent, O. Delalleau, N. Le Roux, and M. Ouimet. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. *Advances in neural information processing systems*, 16:177–184, 2004.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. *NIPS*, 19:153, 2007.

Y. Bengio, A. Courville, and P. Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *Arxiv preprint arXiv:1206.5538*, 2012.

A.C. Berg, T.L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 26–33. IEEE, 2005.

M. Blum, J.T. Springenberg, J. Wulfing, and M. Riedmiller. A learned feature descriptor for object recognition in rgb-d data. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1298–1303. IEEE, 2012.

L. Bo, K. Lai, X. Ren, and D. Fox. Object recognition with hierarchical kernel descriptors. In *CVPR)*, pages 1729–1736. IEEE, 2011a.

L. Bo, X. Ren, and D. Fox. Hierarchical Matching Pursuit for Image Classification: Architecture and Fast Algorithms. In *Advances in Neural Information Processing Systems*, December 2011b.

L. Bo, X. Ren, and D. Fox. Unsupervised feature learning for rgb-d based object recognition. ISER, 2012a.

L. Bo, X. Ren, and D. Fox. Unsupervised Feature Learning for RGB-D Based Object Recognition. In *ISER*, June 2012b.

Anna Bosch, Andrew Zisserman, and Xavier Munoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 401–408. ACM, 2007.

J. Bromley, J.W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, and R. Shah. Signature verification using a "siamese" time delay neural network. *IJPRAI*, 7(4):669–688, 1993.

J. Bruna and S. Mallat. Invariant scattering convolution networks. 2012.

L. Cayton. Algorithms for manifold learning. *University of California, San Diego, Tech. Rep. CS2008-0923*, 2005.

D. Cireşan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. *Arxiv preprint arXiv:1202.2745*, 2012.

Dan Ciresan, Alessandro Giusti, Juergen Schmidhuber, et al. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in Neural Information Processing Systems 25*, pages 2852–2860, 2012.

D.C. Cireşan, U. Meier, J. Masci, L.M. Gambardella, and J. Schmidhuber. High-performance neural networks for visual object classification. *Arxiv preprint arXiv:1102.0183*, 2011.

A. Coates and A.Y. Ng. The importance of encoding versus training with sparse coding and vector quantization. In *ICML*, volume 8, page 10, 2011a.

A. Coates and A.Y. Ng. Selecting receptive fields in deep networks. NIPS, 2011b.

A. Coates, H. Lee, and A. Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011.

Adam Coates and Andrew Y Ng. The importance of encoding versus training with sparse coding and vector quantization. In *International Conference on Machine Learning*, volume 8, page 10, 2011c.

JA Cook, I. Sutskever, A. Mnih, and GE Hinton. Visualizing similarity data with a mixture of maps. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics*, volume 2, pages 67–74. Citeseer, 2007.

C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

A. Criminisi, D. Robertson, E. Konukoglu, J. Shotton, S. Pathak, S. White, and K. Siddiqui. Regression forests for efficient anatomy detection and localization in computed tomography scans. *Medical Image Analysis*, 2013.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886 –893 vol. 1, june 2005. doi: 10.1109/CVPR.2005.177.

L. Deng and D. Yu. Deep convex net: A scalable architecture for speech pattern classification. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

SL Feng, R. Manmatha, and V. Lavrenko. Multiple bernoulli relevance models for image and video annotation. In *CVPR*, volume 2, pages II–1002. IEEE, 2004.

A. Frangi, W. J. Niessen, K. L. Vincken, and M. A. Viergever. Multiscale vessel enhancement filtering. In *MICCAI*, pages 130–137, 1998.

A. Globerson and S. Roweis. Metric learning by collapsing classes. *NIPS*, 18:451, 2006.

X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *JMLR W&CP: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*, 2011.

J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. 2004.

M. Grubinger, P. Clough, H. Müller, and T. Deselaers. The iapr tc-12 benchmark: A new evaluation resource for visual information systems. In *International Workshop OntoImage*, pages 13–23, 2006.

M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, pages 309–316. Ieee, 2009.

R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 1735–1742. IEEE, 2006.

P. Hamel and D. Eck. Learning features from music audio with deep belief networks. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, 2010.

P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In *the 12th International Symposium on Experimental Robotics (ISER)*, 2010.

G. Hinton and S. Roweis. Stochastic neighbor embedding. *Advances in neural information processing systems*, 15:833–840, 2002.

G. Hinton, A. Krizhevsky, and S. Wang. Transforming auto-encoders. *Artificial Neural Networks and Machine Learning–ICANN 2011*, pages 44–51, 2011.

G.E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

G.E. Hinton and R.R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

G.E. Hinton, S. Osindero, and Y.W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R.R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *Arxiv preprint arXiv:1207.0580*, 2012.

G. B. Huang, H. Lee, and E. Learned-Miller. Learning hierarchical representations for face verification with convolutional deep belief networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2518–2525, 2012.

M.J. Huiskes and M.S. Lew. The mir flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 39–43. ACM, 2008.

K. Jarrett, K. Kavukcuoglu, M.A. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE, 2009.

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

K. Kavukcuoglu, M.A. Ranzato, and Y. LeCun. Fast inference in sparse coding algorithms with applications to object recognition. *Arxiv preprint arXiv:1010.3467*, 2010a.

K. Kavukcuoglu, P. Sermanet, Y.L. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning convolutional feature hierarchies for visual recognition. *Advances in Neural Information Processing Systems*, pages 1090–1098, 2010b.

D. Keysers, T. Deselaers, C. Gollan, and H. Ney. Deformation models for image recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(8):1422–1435, 2007.

Karl Krissian, Grégoire Malandain, Nicholas Ayache, Régis Vaillant, and Yves Trousset. Model-based detection of tubular structures in 3d images. *Computer vision and image understanding*, 80(2):130–171, 2000.

A. Krizhevsky. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 2010.

K. Labusch, E. Barth, and T. Martinetz. Simple method for high-performance digit recognition based on sparse coding. *IEEE Transactions on Neural Networks*, 19(11):1985–1989, 2008.

K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011.

Christoph H Lampert, Matthew B Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. Ieee, 2006.

Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Semi-local affine parts for object recognition. In *British Machine Vision Conference*, volume volume 2, pages 779–788, 2004. URL `http://lear.inrialpes.fr/pubs/2004/LSP04`.

Q. V. Le, A. Karpenko, J. Ngiam, and A. Y. Ng. ICA with Reconstruction Cost for Efficient Overcomplete Feature Learning. In *Advances in Neural Information Processing Systems*, 2011a.

Q. V. Le, W. Zou, S. Y. Yeung, and A. Y. Ng. Learning hierarchical spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 2011b.

Q.V. Le, R. Monga, M. Devin, G. Corrado, K. Chen, M.A. Ranzato, J. Dean, and A.Y. Ng. Building high-level features using large scale unsupervised learning. *Arxiv preprint arXiv:1112.6209*, 2011c.

B.B. Le Cun, JS Denker, D. Henderson, RE Howard, W. Hubbard, and LD Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*. Citeseer, 1990.

Chi-Hoon Lee, Mark Schmidt, Albert Murtha, Aalo Bistritz, Jöerg Sander, and Russell Greiner. Segmenting brain tumors with conditional random fields and support vector machines. In *Computer Vision for Biomedical Image Applications*, pages 469–478. Springer, 2005.

H. Lee, A. Battle, R. Raina, and A.Y. Ng. Efficient sparse coding algorithms. *Advances in neural information processing systems*, 19:801, 2007.

H. Lee, R. Grosse, R. Ranganath, and A.Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009a.

Honglak Lee, Yan Largman, Peter Pham, and Andrew Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems 22*, pages 1096–1104. 2009b.

E. Levina and P.J. Bickel. Maximum likelihood estimation of intrinsic dimension. *Ann Arbor MI*, 48109:1092, 2004.

L.J. Li and L. Fei-Fei. What, where and who? classifying events by scene and object recognition. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

L.J. Li, H. Su, E.P. Xing, and L. Fei-Fei. Object bank: A high-level image representation for scene classification and semantic feature sparsification. *Advances in Neural Information Processing Systems*, 24, 2010.

D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157. Ieee, 1999.

A. Makadia, V. Pavlovic, and S. Kumar. A new baseline for image annotation. In *ECCV*, volume 8, pages 316–329, 2008.

James Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, volume 951, page 2010, 2010.

C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox. A joint model of language and perception for grounded attribute learning. *Arxiv preprint arXiv:1206.6423*, 2012.

R. Min, D.A. Stanley, Z. Yuan, A. Bonner, and Z. Zhang. A deep non-linear feature mapping for large-margin knn classification. In *Ninth IEEE International Conference on Data Mining*, pages 357–366. IEEE, 2009.

R. Min, L. van der Maaten, Z. Yuan, A. Bonner, and Z. Zhang. Deep supervised t-distributed embedding. In *ICML*, volume 1, 2010.

A. Mohamed, G. Dahl, and G. Hinton. Deep belief networks for phone recognition. In *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*, 2009.

V. Nair and G.E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proc. 27th International Conference on Machine Learning*, 2010.

H. Nakayama. *Linear Distance Metric Learning for Large-scale Generic Image Recognition*. PhD thesis, The University of Tokyo, 2011.

Jiquan Ngiam, Pang Wei W. Koh, Zhenghao Chen, Sonia A. Bhaskar, and Andrew Y. Ng. Sparse filtering. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1125–1133. 2011.

A. Oliva. Gist of the scene. *Neurobiology of attention*, pages 251–256, 2005.

M. Pandey and S. Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1307–1314. IEEE, 2011.

N. Paragios and R. Deriche. Geodesic active regions and level set methods for supervised texture segmentation. *IJCV*, 50(3):223–247, 2002.

Simon Perkins, Kevin Lacker, and James Theiler. Grafting: Fast, incremental feature selection by gradient descent in function space. *The Journal of Machine Learning Research*, 3:1333–1356, 2003.

A. Quattoni and A. Torralba. Recognizing indoor scenes. 2009.

R. Raina, A. Battle, H. Lee, B. Packer, and A.Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766. ACM, 2007.

M. Ranzato, C.P., S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. *NIPS*, 19:1137–1144, 2006.

M.A. Ranzato, F.J. Huang, Y.L. Boureau, and Y. Lecun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. Ieee, 2007.

M.A. Ranzato, J. Susskind, V. Mnih, and G. Hinton. On deep generative models with applications to recognition. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2857–2864. IEEE, 2011.

S. Rifai, G. Mesnil, P. Vincent, X. Muller, Y. Bengio, Y. Dauphin, and X. Glorot. Higher order contractive auto-encoder. *Machine Learning and Knowledge Discovery in Databases*, pages 645–660, 2011a.

Salah Rifai, Yann N. Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller. The manifold tangent classifier. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2294–2302. 2011b.

Roberto Rigamonti and Vincent Lepetit. Accurate and efficient linear structure. segmentation by leveraging ad hoc features with learned filters. In *MICCAI*, pages 189–197, 2012.

Roberto Rigamonti, Engin Türetken, Germán González, Pascal Fua, and Vincent Lepetit. Filter learning for linear structure segmentation. Technical report, Technical report, EPFL, 2011.

S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

R. Rubinstein, M. Zibulevsky, and M. Elad. Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit. *CS Technion*, 2008.

D.E. Rumelhart, G.E. Hintont, and R.J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

R. Salakhutdinov and G. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *AI and Statistics*, volume 3, page 5, 2007a.

R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

Ruslan Salakhutdinov and Geoffrey Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *AI and Statistics*, volume 3, page 5, 2007b.

J.W. Sammon Jr. A nonlinear mapping for data structure analysis. *Computers, IEEE Transactions on*, 100(5):401–409, 1969.

F. Seide, G. Li, and D. Yu. Conversational speech transcription using context-dependent deep neural networks. In *Twelfth Annual Conference of the International Speech Communication Association*, 2011.

P. Sermanet, S. Chintala, and Y. LeCun. Convolutional neural networks applied to house numbers digit classification. *Arxiv preprint arXiv:1204.3968*, 2012.

J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, volume 2, page 7, 2011.

Kihyuk Sohn, Dae Yon Jung, Honglak Lee, and Alfred Hero III. Efficient learning of sparse, distributed, convolutional feature representations for object recognition. In *Proceedings of 13th International Conference on Computer Vision*, 2011.

M. Strickert, A.J. Soto, and G.E. Vazquez. Adaptive matrix distances aiming at optimum regression subspaces. In *ESANN*, pages 93–98, 2010.

G.W. Taylor, R. Fergus, G. Williams, I. Spiro, and C. Bregler. Pose-sensitive embedding by non-linear nca regression.

J.B. Tenenbaum, V. De Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *CVPR*, pages 1–8. IEEE, 2008.

D. Tsai, Y. Jing, Y. Liu, H.A. Rowley, S. Ioffe, and J.M. Rehg. Large-scale image annotation using visual synset. In *ICCV*, pages 611–618. IEEE, 2011.

L. van der Maaten. Learning a parametric embedding by preserving local structure. *Proceedings of AI-STATS*, 2009.

L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

L. van der Maaten and G. Hinton. Visualizing non-metric similarities in multiple maps. *Machine Learning*, pages 1–23, 2011.

P. Vincent, H. Larochelle, Y. Bengio, and P.A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

L. Von Ahn and L. Dabbish. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 319–326. ACM, 2004.

Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3360–3367. IEEE, 2010.

Yang Wang and Greg Mori. Max-margin latent dirichlet allocation for image classification and annotation. In *22nd British Machine Vision Conference (BMVC)*, 2011.

K.Q. Weinberger and L.K. Saul. Unsupervised learning of image manifolds by semidefinite programming. In *CVPR*, volume 2, pages II–988. IEEE, 2004.

K.Q. Weinberger, J. Blitzer, and L.K. Saul. Distance metric learning for large margin nearest neighbor classification. In *In NIPS*. Citeseer, 2006.

J. Weston, F. Ratle, and R. Collobert. Deep learning via semi-supervised embedding. In *ICML*, pages 1168–1175. ACM, 2008.

J. Weston, S. Bengio, and N. Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning*, 81(1):21–35, 2010.

J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801. Ieee, 2009.

K. Yu, Y. Lin, and J. Lafferty. Learning image representations from the pixel level via hierarchical sparse coding. In *CVPR*, pages 1713–1720. IEEE, 2011.

Y. Yu, J. Neufeld, R. Kiros, X. Zhang, and D. Schuurmans. Regularizers versus losses for non-linear dimensionality reduction: A factored view with new convex relaxations. *Arxiv preprint arXiv:1206.6455*, 2012.

Y. Hu Z. Wang and L.T. Chia. Multi-label learning by image-to-class distance for scene classification and image annotation. In *CIVR*, pages 105–112. ACM, 2010.

M.D. Zeiler and R. Fergus. Differentiable pooling for hierarchical feature learning. *Arxiv preprint arXiv:1207.0151*, 2012.

M.D. Zeiler, G.W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2018–2025. IEEE, 2011.

M.L. Zhang and Z.H. Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recognition*, 40(7):2038–2048, 2007.

M.L. Zhang and Z.H. Zhou. Multi-label learning by instance differentiation. In *National Conference on Artifical Intelligence*, volume 22, page 669. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

S. Zhang, J. Huang, Y. Huang, Y. Yu, H. Li, and D.N. Metaxas. Automatic image annotation using group sparsity. In *CVPR*, pages 3312–3319. IEEE, 2010.

X. Zhang, Y. Yu, M. White, R. Huang, and D. Schuurmans. Convex sparse coding, subspace learning, and semi-supervised extensions. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

Y. Zheng, A. Barbu, B. Georgescu, M. Scheuering, and D. Comaniciu. Four-chamber heart modeling and automatic segmentation for 3d cardiac ct volumes using marginal space learning and steerable features. *IEEE Trans. Medical Imaging*, 27(11):1668–1681, 2008.

Z.H. Zhou and M.L. Zhang. Multi-instance multi-label learning with application to scene classification. *Advances in Neural Information Processing Systems*, 19:1609, 2007.

J. Zhu, L.J. Li, L. Fei-Fei, and E.P. Xing. Large margin learning of upstream scene understanding models. *Advances in Neural Information Processing Systems*, 24, 2010.

W. Zou, A. Ng, and Kai. Yu. Unsupervised learning of visual invariance with temporal coherence. In *NIPS*, 2011.