Learning representations for visual recognition

Shreyas Saxena

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Mathématiques, Sciences et Technologies de l'Information**

Arrêté ministériel : 7 août 2006

Présentée par

Shreyas Saxena

Thèse dirigée par **Cordelia SCHMID**
et codirigée par **Jakob Verbeek**

préparée au sein d' **Inria Grenoble**
et de **l'école doctorale MSTII : Mathématiques, Sciences et Technologies de l'Information, Informatique**

Learning representations for visual recognition

Apprentissage de représentations pour la reconnaissance visuelle

Thèse soutenue publiquement le **12 Décembre 2016**,
devant le jury composé de :

Prof. Frederic Jurie
University of Caen, Caen, France, Rapporteur
Prof. Tinne Tuytelaars
Katholieke Universiteit Leuven, Leuven, Belgium, Rapporteur
Dr. Andrew Bagdanov
University of Florence, Florence, Italy, Examinateur
Dr. Cordelia Schmid
Inria Grenoble, Montbonnot, France, Directeur de thèse
Dr. Jakob Verbeek
Inria Grenoble, Montbonnot, France, Co-Directeur de thèse

Abstract

In this dissertation, we propose methods and data driven machine learning solutions which address and benefit from the recent overwhelming growth of digital media content.

First, we consider the problem of improving the efficiency of image retrieval. We propose a coordinated local metric learning (CLML) approach which learns local Mahalanobis metrics, and integrates them in a global representation where the ℓ_2 distance can be used. This allows for data visualization in a single view, and use of efficient ℓ_2-based retrieval methods. Our approach can be interpreted as learning a linear projection on top of an explicit high-dimensional embedding of a kernel. This interpretation allows for the use of existing frameworks for Mahalanobis metric learning for learning local metrics in a coordinated manner. Our experiments show that CLML improves over previous global and local metric learning approaches for the task of face retrieval.

Second, we present an approach to leverage the success of CNN models for visible spectrum face recognition to improve heterogeneous face recognition, *e.g.*, recognition of near-infrared images from visible spectrum training images. We explore different metric learning strategies over features from the intermediate layers of the networks, to reduce the discrepancies between the different modalities. In our experiments we found that the depth of the optimal features for a given modality, is positively correlated with the domain shift between the source domain (CNN training data) and the target domain. Experimental results show the that we can use CNNs trained on visible spectrum images to obtain results that improve over the state-of-the art for heterogeneous face recognition with near-infrared images and sketches.

Third, we present convolutional neural fabrics for exploring the discrete and exponentially large CNN architecture space in an efficient and systematic manner. Instead of aiming to select a single optimal architecture, we propose a "fabric" that embeds an exponentially large number of architectures. The fabric consists of a 3D trellis that connects response maps at different layers, scales, and channels with a sparse homogeneous local connectivity pattern. The only hyper-parameters of the fabric (the number of channels and layers) are not critical for performance. The acyclic nature of the fabric allows us to use backpropagation for learning. Learning can thus efficiently configure the fabric to implement each one of exponentially many architectures and, more generally, ensembles of all of them. While scaling linearly in terms of computation and memory requirements, the fabric leverages exponentially many chain-structured architectures in parallel by massively sharing weights between them. We present benchmark results competitive with the state of the art for image classification on MNIST and CIFAR10, and for semantic segmentation on the Part Labels dataset.

Keywords. Local metric learning • Transfer learning • Convolutional neural network • Architecture learning

Résumé

Dans cette dissertation, nous proposons des méthodes d'apprentissage automatique aptes à bénéficier de la récente explosion des volumes de données digitales.

Premièrement nous considérons l'amélioration de l'efficacité des méthodes de récupération d'image. Nous proposons une approche d'apprentissage de métriques locales coordonnées (Coordinated Local Metric Learning, CLML) qui apprends des métriques locales de Mahalanobis, puis les intègre dans une représentation globale où la distance l2 peut être utilisée. Ceci permet de visualiser les données avec une unique représentation 2D, et l'utilisation de méthodes de récupération efficaces basées sur la distance l2. Notre approche peut être interprétée comme l'apprentissage d'une projection linéaire de descripteurs donnés par une méthode a noyaux de grande dimension définie explictement. Cette interprétation permet d'appliquer des outils existants pour l'apprentissage de métriques de Mahalanobis à l'apprentissage de métriques locales coordonnées. Nos expériences montrent que la CLML améliore les résultats en matière de récupération de visage obtenues par les approches classiques d'apprentissage de métriques locales et globales.

Deuxièmement, nous présentons une approche exploitant les modèles de réseaux neuronaux convolutionnels (CNN) pour la reconnaissance faciale dans le spectre visible. L'objectif est l'amélioration de la reconnaissance faciale hétérogène, c'est à dire la reconnaissance faciale à partir d'images infra-rouges avec des images d'entraînement dans le spectre visible. Nous explorerons différentes stratégies d'apprentissage de métriques locales à partir des couches intermédiaires d'un CNN, afin de faire le rapprochement entre des images de sources différentes. Dans nos expériences, la profondeur de la couche optimale pour une tâche donnée est positivement corrélée avec le changement entre le domaine source (données d'entraînement du CNN) et le domaine cible. Les résultats montrent que nous pouvons utiliser des CNN entraînés sur des images du spectre visible pour obtenir des résultats meilleurs que l'état de l'art pour la reconnaissance faciale hétérogène (images et dessins quasi-infrarouges).

Troisièmement, nous présentons les "tissus de neurones convolutionnels" (Convolutional Neural Fabrics) permettant l'exploration de l'espace discret et exponentiellement large des architectures possibles de réseaux neuronaux, de manière efficiente et systématique. Au lieu de chercher à sélectionner une seule architecture optimale, nous proposons d'utiliser un "tissu" d'architectures combinant un nombre exponentiel d'architectures en une seule. Le tissu est une représentation 3D connectant les sorties de CNNs à différentes couches, échelles et canaux avec un motif de connectivité locale, homogène et creux. Les seuls hyper-paramètres du tissu (le nombre de canaux et de couches) ne sont pas critiques pour la performance. La nature acyclique du tissu nous permet d'utiliser la rétro-propagation du gradient durant la phase d'apprentissage. De manière automatique, nous pouvons donc configurer le tissu de manière à implémenter l'ensemble de toutes les architectures possibles (un nombre exponentiel) et, plus généralement, des ensembles (combinaisons) de ces

modèles. La complexité de calcul et de taille mémoire du tissu évoluent de manière linéaire alors qu'il permet d'exploiter un nombre exponentiel d'architectures en parallèle, en partageant les paramètres entre architectures. Nous présentons des résultats à l'état de l'art pour la classification d'images sur le jeu de données MNIST et CIFAR10, et pour la segmentation sémantique sur le jeu de données Part Labels.

Mots-clés. Apprentissage de métriques locales • Transfert d'apprentissage • Réseaux neuronaux convolutionnels • Apprentissage d'architectures.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

The Cambrian explosion refers to the short evolutionary event beginning around 500 million years ago, which witnessed sudden appearance of fossil record of major animal groups. A similar explosion is currently under process in the digital age.

> "Between the dawn of civilization and 2003, we only created five exabytes; now we're creating that amount every two days. By 2020, that figure is predicted to sit at 53 zettabytes (53 trillion gigabytes) – an increase of 50 times."
>
> —HAL VARIAN, Chief Economist, Google

A significant part of this data explosion is due to the increase in the digital media. The number and quality of capturing devices have significantly grown over the past years, making it possible for us to collect large amount of personal multimedia collections. The widespread availability of internet and social platforms such as Flickr, Facebook, Instagram, and Youtube, allows us to access and share the multimedia collections, creating a vast storage of digital data in the cloud. To get a perspective, in the one minute you have spent reading the thesis so far, 400 hours of video has been uploaded on Youtube [1], 136,000 photos have been uploaded on Facebook [2], and 3,600 photos have been shared on Instagram [3]. An-

1. https://www.youtube.com/
2. https://www.facebook.com/
3. https://www.instagram.com/

other major source of increase in the digital media is surveillance. For instance, 413 Petabytes of data is roughly four times the amount of photos and videos stored on Facebook [4]. Surprisingly, this is also the amount of data produced in *one day* by all cameras installed worldwide in 2013 [5].

The rapid increase in the digital content comes with both, challenges and benefits. On one hand, storing vast amounts of data has lead to an increase in the storage and computational footprint, and requires efficient storage and indexing methods. On the other hand, the abundance of data can be used for training statistical learning models.

1.1 Context

The proliferation of the digital data has made a significant impact on the computer vision and the machine learning community. In the following paragraphs, we list the two outcomes of the rapid increase in digital media, which are of interest for this dissertation.

The first consequence is the need for efficient systems to process this large volume for different tasks. Traditional database techniques for processing a large database have been adequate for applications including text records which could be ordered, indexed and queried for matching patterns in a straightforward manner. In contrast, for non-textual unstructured content such as images, indexing and querying is not as obvious as it is for text records. This is due to the fact that an image first needs to be encoded into a semantically meaningful representation, before it can be indexed in a database and queried.

> " When you think about "information," what probably comes to mind are streams of words and numbers. Google's pretty good at organizing these types of information, but consider all the things you can not express with words: what does it look like in the middle of a sandstorm? What are some great examples of Art Nouveau architecture? Should I consider wedding cupcakes instead of a traditional cake? This is why we built Google Images. "
>
> — NATE SMITH, Product Manger, Google Images

4. As of February 2012
5. http://press.ihs.com/press-release/design-supply-chain-media/big-big-big-data-rise-hd-video-surveillance-cameras-spurs-in

Indexing multimedia data based on its content simplifies its manageability and leads to efficiency in search. For example, Google Images was launched in 2001 with around 250 million images indexed. By 2010, the amount of indexed pictures has grown to an astounding 10 billion in number. Dealing with images at a large scale makes indexing and retrieval a challenging and an important task. As per Rui et al. [1999], the three fundamental components for content-based image retrieval are: (i) visual feature extraction, (ii) multidimensional indexing and (iii) retrieval system design. In the recent decade a lot of progress has been made on these three components. In particular, the visual feature extraction techniques have evolved from primitive color and gray scale histograms [Swain and Ballard, 1991, Huang et al., 1999] to scale and viewpoint invariant high-dimensional descriptors [Perronnin and Dance, 2007, Jégou et al., 2010, Csurka et al., 2004]. Recent progress has been made using supervised learning and efficient quantization to improve the accuracy of retrieval systems, and to address the computational and storage requirements. Progress along these lines has been vital for scaling up the traditional retrieval systems and will continue to play a vital role in the coming years.

Another consequence of the expansion in digital media is its role in our understanding of images. Statistical learning methods learn the model parameters from large-scale datasets. Once the training is complete these models can be evaluated on previously unseen images. The optimal complexity of these model (in terms of parameters) is largely determined by the number of available labelled images. Recently, the availability of large scale datasets [Deng et al., 2009] have facilitated the impressive results obtained with Convolutional Neural Networks (CNNs) [LeCun et al., 1989]. In particular, the recent results of Krizhevsky et al. [2012] have caused a major paradigm shift in the computer vision community, from models relying on hand-crafted features to end-to-end trainable systems. However, this recent success of representation learning comes with few limitations. First, training a deep CNN requires a large amount of labelled data, and hence the application of CNN to datasets with a small sample size is not straightforward. To mitigate this issue, recent works have focused on transferring the learned representations across a variety of datasets and tasks [Sharif Razavian et al., 2014]. Transferability of these learned representations continues to be one of the important research directions in the community.

The recent success of end-to-end trainable systems has made the practice of hand-crafting image descriptors obsolete in computer vision community. However, this success came at the cost of introducing a new practice in the community *i.e*. hand-crafting network architectures. This is the second limitation with

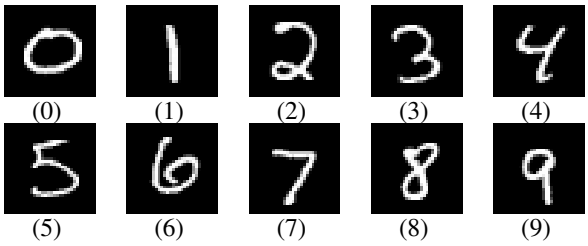


Figure 1.1 – Illustration for image classification. In the figure, we show some image instances representing different digits with their corresponding labels.

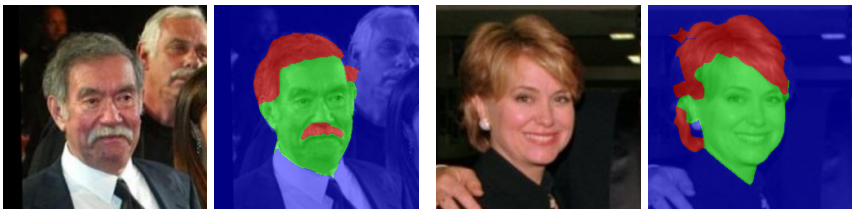


Figure 1.2 – Examples form the Part Labels dataset for semantic segmentation. The image pixels are classified into three classes: hair, background and skin.

the CNN based approaches, and is concerned with the large number of hyper-parameters which needs to set for obtaining a network architecture. At present, in the computer vision community these hyperparameters are set manually, *i.e*. hand-crafted for a particular dataset or task.

1.2 Goals

The focus of this thesis is on methods and data driven machine learning solutions which address and benefit from consequences of rapid increase in digital media. In the following, we briefly describe the vision problems considered in the dissertation.

Image classification The task of image classification is to assign an input image to one of the given classes based on their visual content, see Figure 1.1. We distinguish between multiclass classification, where images are associated with a single label, and multi-label classification, where an image can be related to more than one label.

Image segmentation The task of image segmentation is similar to image classification, but involves making predictions at a finer scale. It is similar to classification in the sense that it involves classifying each pixel of an input image to one of the given classes, see Figure 1.2.

Figure 1.3 – Pairs corresponding to the two categories of face verification, namely, match pairs (left) and mismatch pairs (right).

Face recognition Face recognition is mainly divided into two tasks: (i) face identification and (ii) face verification. The goal of face identification is to assign the input image to one of the several predefined identities in a database. In contrast, the goal of face verification is to verify whether the two input images belong to the same identity or not, see Figure 1.3 for illustration.

Face retrieval Compared to face verification, face retrieval is related but a slightly different task. The user provides the face of a person as a query, and the goal is to retrieve images in a database which depict that person.

Heterogeneous face recognition The goal of heterogeneous face recognition is to recognize faces of people across different modalities. In most cases, the gallery of known individuals consists of normal visible spectrum images. Probe images may be forensic sketches or thermal infrared images which are useful in a forensic context or covert non-intrusive night-time acquisition respectively, see Figure 1.4.

1.3 Contributions

The following paragraphs explain the problems that we focus on in this thesis, and our corresponding contributions.

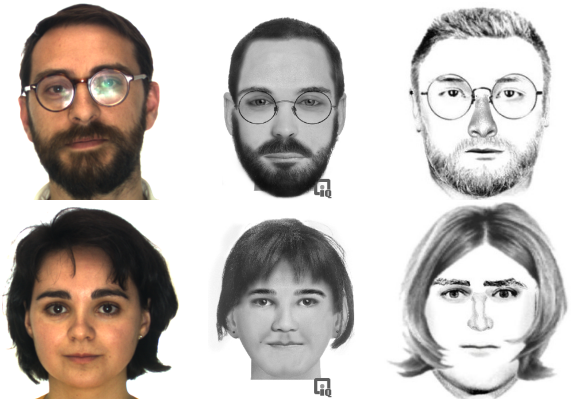


Figure 1.4 – Example images from E-PRIP database which depict the modality gap between visible spectrum images and sketches.

What are the limitations of metric learning approaches, and how can we address them? Mahalanobis metric learning amounts to learning a linear data projection, after which the ℓ_2 metric is used to compute distances. To allow more flexible metrics, not restricted to linear projections, local metric learning techniques have been developed. Most of these methods partition the data space using clustering, and for each cluster a separate metric is learned. Using local metrics, however, it is not clear how to measure distances between data points assigned to different clusters. In our work, we propose to embed the local metrics in a global representation. This global representation directly allows computing distances between points regardless to which local cluster they belong. Moreover, it also enables data visualization in a single view (see Figure 1.5), and the use of ℓ_2-based efficient retrieval methods. In our work, we show that our approach for learning local metrics with alignment can be equivalently interpreted as learning a linear projection on an explicit feature embedding of a high dimensional kernel. This is the main contribution of work, which allows the use of existing global Mahalanobis metric learning methods for learning coordinated local metrics. Experiments on the Labeled Faces in the Wild dataset show that our approach improves over previous global and local metric learning approaches for the task of face retrieval. Our approach also leads to larger speed-ups at a higher level of performance. This work is published in [Saxena and Verbeek, 2015] and presented in Chapter 2.

How can we leverage the success of CNN models for visible spectrum face recognition to improve heterogeneous face recognition? Heterogeneous face recognition aims to recognize faces across different sensor modalities. Typically, gallery images are normal visible spectrum images, and probe images are

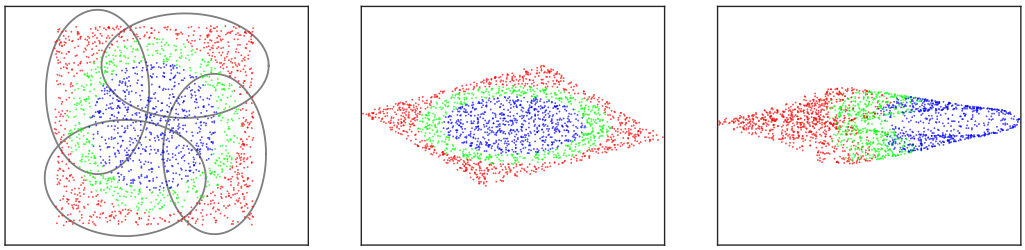


Figure 1.5 – Synthetic dataset with color coded class labels, and the local clusters used by our method (left panel). Data projection given by a global Mahalanobis metric (center panel) and our coordinated local metrics (right panel). The latter approximately separates the classes via local linear maps.

infrared images or sketches, see Figure 1.4. Recently significant improvements in visible spectrum face recognition have been obtained by CNNs learned from very large training datasets. It is, however, not straightforwad how to apply such networks for heterogeneous face recognition, since the image characteristics may differ significantly across the modalities, and typically relatively little training data is available for other modalities than normal visible spectrum images. In this work, we are interested in the question to what extent the features from a CNN pre-trained on visible spectrum face images can be used to perform heterogeneous face recognition. We explore different metric learning strategies to reduce the discrepancies between the different modalities and find that metric learning over features of intermediate layers of the network is most effective. Experimental results show that we can use CNNs trained on visible spectrum images to obtain results that are on par or improve over the state-of-the-art for heterogeneous recognition with near-infrared images and sketches. This work is published in [Saxena and Verbeek, 2016b] and presented in Chapter 3.

How can we explore the discrete and exponentially large architecture space of a CNN in an efficient and systematic manner? The architecture of standard CNNs is determined by several hyperparameters, including, but not limited to the following: number of layers, number of channels per layer, filter size per layer and number of pooling *vs*. convolutional layers . Despite the success of CNNs, selecting the optimal architecture for a given task remains an open problem. Instead of aiming to select a single optimal architecture, we propose a "fabric" that embeds an exponentially large number of architectures. The fabric consists of a 3D trellis that connects response maps at different layers, scales, and channels with a sparse homogeneous local connectivity pattern, see Figure 1.6. The only hyper-parameters of a fabric are the number of channels and layers. The standard classification and segmentation architectures can be recov-

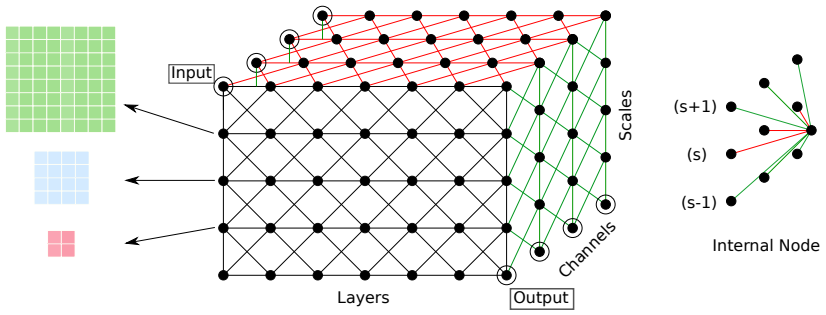


Figure 1.6 – On the left, schematic illustration of the sparse homogeneous edge structure in the trellis. Channel connectivity pattern at the same scale (red) and across finer and coarser scale (green) depict sparse connect structure. On the right, connectivity of a channel of an internal node to channels of preceding nodes is shown. The internal node is connected to 3 channels at the same scale (red), 3 at the finer and 3 at coarser scales (green).

ered from the fabric by setting certain weights to zero, so that only a single activation path is non-zero across the fabric. The acyclic nature of the fabric allows us to use backpropogation for learning. Learning can thus efficiently configure the fabric to implement each one of exponentially many architectures and, more generally, ensembles of all of them. An attractive property of the fabrics is its low computational complexity for exploration of the architecture space. While scaling linearly in terms of computation and memory requirements, the fabric leverages exponentially many chain-structured architectures in parallel by massively sharing weights between them. We present benchmark results competitive with the state of the art for image classification on MNIST and CIFAR10, and for semantic segmentation on the Part Labels dataset. This work is published in [Saxena and Verbeek, 2016a] and presented in Chapter 4.

The structure of the thesis is as follows. We present our technical contributions in Chapter 2, 3 and 4. The overview of the related work relevant for each contribution is presented in their respective chapters. We conclude the thesis with a summary and perspectives in Chapter 5.

Chapter 2

Coordinated Local Metric Learning

Contents

2.1 Introduction

Many supervised and unsupervised machine learning problems are based on a notion of metric (similarity or distance function) between examples. Metrics play a crucial role in a wide range of applications in computer vision, *e.g.* local descriptor matching (Dosovitskiy et al. [2014]), fine-grained object comparison

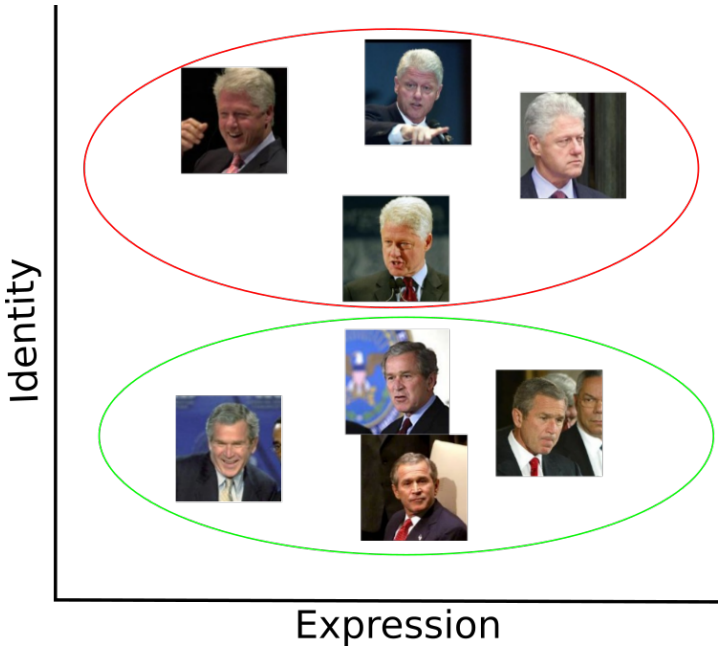


Figure 2.1 – Face images represented by two fictious dimensions, identity and expression. The ellipses depict the subject-dependent variability in the feature space. One can see from the figure, that depending on the task different metrics are needed. See text for details.

(Nowak and Jurie [2007]), image segmentation (Lajugie et al. [2014]) , and face verification (Köstinger et al. [2012]).

The performance of algorithms in these problems depends upon how relevant is the metric used to the task at hand. For instance, we hope that the distance between instances that share a label is smaller compared to the distance between instances which do not share the label. Unfortunately, standard distance metrics like Euclidean distance computed over the high-dimensional features, often fail to capture the specific nature of the task at hand.

Need of metric learning As discussed before, need of metric learning arises when we need different metrics tailored for different tasks. We motivate this need through a toy example in Figure 2.1. The face images are represented in a fictious 2 dimensional coordinate system. First dimension corresponds to identity of the person, while the second corresponds to expression. In this particular descriptor space, if our goal was to design an ideal metric for the the task of face verification, then the metric should highlight the identity dimension and suppress the expression dimension. Doing so, will make the resulting metric invariant to

expression changes in the face. However, if we were to design an ideal metric for the task of expression recognition, then the metric should suppress the identity dimension, making the faces of two people with same expression more similar.

This toy example exemplifies the need for metric learning *i.e*. for a given input space, different tasks require a different metric. In the example above, we had fictious coordinate axes, but when dealing with real data (for *e.g*. face images represented by vectorized pixel values) we do not know which dimensions are crucial for the task at hand. This is the precise goal of metric learning, where we learn a metric relevant to the problem at hand using the information brought by a sample of labeled examples.

In the past years, a considerable amount of research effort has been devoted to design and learn metrics from the data. Most work considers supervised learning of Mahalanobis metrics, see *e.g*. (Globerson and Roweis [2006], Davis et al. [2007], Guillaumin et al. [2009], Weinberger and Saul [2009], Köstinger et al. [2012], Mignon and Jurie [2012b], Shi et al. [2014], Trivedi et al. [2014]).

The supervision comes as positive and negative pairs that should be close and far apart respectively. The Mahalanobis distance between two points is given by $(\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{M}(\mathbf{x}_i - \mathbf{x}_j)$, where $\mathbf{M}$ is a positive definite matrix. Since $\mathbf{M}$ can always be factored as $\mathbf{M} = \mathbf{L}^\top \mathbf{L}$, Mahalanobis metrics are equivalent to the ℓ_2 metric after linear projection of the data. For complex class distributions, however, linear projection of the data might not be sufficient to obtain a suitable data representation.

To overcome this restriction, several routes have been explored. First, the linear projection in the Mahalanobis metric can be written in terms of kernel evaluations, see *e.g*. (Globerson and Roweis [2006], Guillaumin et al. [2010], Mignon and Jurie [2012b]). Alternatively, (convolutional) neural networks with a siamese architecture can be learned to give (dis)similar outputs for positive and negative pairs, see *e.g*. (Bromley et al. [1993], Chopra et al. [2005]). Finally, local metric learning uses a collection of Mahalanobis metrics, each operating in a different part of the input space, see *e.g*. (Bhattarai et al. [2014], Bohné et al. [2014], Frome et al. [2007], Hong et al. [2011], Huang et al. [2013], Noh et al. [2010], Shi et al. [2014], Wang et al. [2012], Weinberger and Saul [2009], Zhan et al. [2009a]). The partitioning of the space is typically obtained using k-means or Gaussian mixture clustering.

In most existing local metric learning approaches, however, it is unclear how to compute distances between points assigned to different clusters, or distances are defined in an asymmetric manner. Unlike for global metric learning, they can not be interpreted as computing the ℓ_2 distance after a transformation of the data, which hinders data visualization and efficient ℓ_2-based retrieval techniques, such as product quantization and multiple-assignment retrieval (Jégou et al. [2011]).

In this chapter we propose a solution by embedding the local metrics in a global representation. In particular, we map the input data to an expanded data representation. This representation is essentially a non-linear embedding of the input data in the high dimensional space, but as we will show this representation does not need to be explicitly constructed, and avoids excessive memory consumption by our method. We show that learning a Mahalanobis metric over the proposed embedding is equivalent to simultaneously learning the local metrics for each cluster, along with their alignment. Therefore, existing Mahalanobis metric learning methods can be used to learn and coordinate local Mahalanobis metrics.

The local metrics are aligned in the global embedding space due to the pairwise constraints imposed by the objective function. The alignment of the local metrics, ensures that the local metrics are coherently placed in a global coordinate system. This allows us to (i) compute distances between points regardless to which local cluster they belong, (ii) visualize data in a single view, and (iii) use efficient ℓ_2-based retrieval methods. We refer to our approach as "coordinated local metric learning" (CLML).

We validate our approach in face verification and retrieval settings using the Labeled Faces in the Wild (LFW) (Huang et al. [2007]) dataset, which has drawn a significant amount of interest over the recent years, see *e.g.* (Bhattarai et al. [2014], Bohné et al. [2014], Cao et al. [2013], Chopra et al. [2005], Guillaumin et al. [2009, 2010], Köstinger et al. [2012], Liao et al. [2014], Mignon and Jurie [2012b], Nowak and Jurie [2007], Simonyan et al. [2013a], Sun et al. [2014], Taigman et al. [2014]). In face verification, the task is to determine whether the two face images depict the same person or not. Face retrieval is a related but slightly different task, the user provides the face of a person as a query, and the goal is to retrieve images in a database which depict that person.

In our experiments we represent the face images using image representations based on local binary patterns (LBP, Ojala et al. [2002]), convolutional neural networks (CNN, Yi et al. [2014]), and Fisher vectors (FV, Simonyan et al.

[2013a]). We use logistic discriminant metric learning method (LDML, Guillaumin et al. [2009]) to learn both global and local metrics and compare their performance. For all tested image representations our approach improves over global metric learning and other local metric learning approaches. For retrieval, the improvements over previous local metric learning approaches (Bhattarai et al. [2014], Shi et al. [2014]) are particularly large.

Outline The rest of the chapter is organized as follows. In Section 2.2 we discuss a selection of related work which is most relevant to the topics of this chapter. In Section 2.3 we present our work on coordinated local metric learning along with implementation details. We present extensive experimental results in Section 2.4, analyzing different aspects of the proposed method and comparing it to the current state-of-the-art in different application settings such as face verification and face retrieval. Finally, in Section 2.5 we conclude this chapter.

2.2 Related work

There has been considerable amount of research in the field of distance metric learning over the past years. Depending on the availability of the labels for training data, the algorithms for distance metric learning can be grouped into two broad categories: supervised and unsupervised. In contrast to most supervised learning algorithms which require training data to have class labels, majority of supervised distance metric learning algorithms require the training data as pairs, which are labelled as positive or negative depending whether the instances in the pair share their class labels or not.

In this section, first we review some basic terminology for distance metric learning, followed by related work on supervised metric learning methods that is most relevant to the material we present in this chapter.

2.2.1 Mahalanobis metrics

A mapping $D : X \times X \rightarrow \mathbb{R}^+$ over a vector space X is called a metric if for all vectors $\forall \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k \in X$, it satisfies the properties:

1. $D(\mathbf{x}_i, \mathbf{x}_j) + D(\mathbf{x}_j, \mathbf{x}_k) \geq D(\mathbf{x}_i, \mathbf{x}_k)$ (triangular inequality).
2. $D(\mathbf{x}_i, \mathbf{x}_j) \geq 0$ (non-negativity).
3. $D(\mathbf{x}_i, \mathbf{x}_j) = D(\mathbf{x}_j, \mathbf{x}_i)$ (symmetry).
4. $D(\mathbf{x}_i, \mathbf{x}_j) = 0 \iff \mathbf{x}_i = \mathbf{x}_j$ (distinguishability).

If a mapping satisfies the first three properties but not the fourth, it is called a pseudometric. However, to simplify the discussion in what follows, we will refer to pseudometrics as metrics, pointing out the distinction only when necessary.

Given the vectorial representation $\mathbf{x}_i \in \mathbb{R}^D$ of data points, where sub-script i indexes the data point, we seek to design good metrics for a given task, *e.g.* verification, retrieval, ranking *etc*. An illustration of metric learning is shown in Figure 2.2. We obtain a family of metrics by computing Euclidean distances after projecting the data with a linear transformation $\tilde{\mathbf{x}}_i = \mathbf{L}\mathbf{x}_i$. The pairwise squared distance is given by,

$$d_{\mathbf{L}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{L}\mathbf{x}_i - \mathbf{L}\mathbf{x}_j)^\top (\mathbf{L}\mathbf{x}_i - \mathbf{L}\mathbf{x}_j). \tag{2.1}$$

The distance in Eq. (2.1) is parametrised by matrix $\mathbf{L}$. If $\mathbf{L}$ is full-rank then the distance defined in Eq. (2.1) is a valid metric otherwise it is a pseudometric. We can equvivalently rewrite the equation as,

$$d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j), \tag{2.2}$$

where $\mathbf{M} = \mathbf{L}^\top \mathbf{L} \in \mathbb{R}^{D \times D}$ and $\mathbf{L} \in \mathbb{R}^{d \times D}$, and d is the rank of $\mathbf{M}$. By construction $\mathbf{M}$ is a symmetric positive semi-definite matrix that parametrises the distance in Eq. (2.2), which is also known as Mahalanobis distance. For $\mathbf{M}$ equal to identity, the Mahalanobis distance reduces to the Euclidean distance. From the equations above, we can see that the Mahalanobis metric corresponds to Euclidean distance after a linear transformation. Therefore, learning the Mahalanobis distance can be equivalently expressed as optimizing either $\mathbf{M}$ or $\mathbf{L}$. It is interesting to note that when the matrix $\mathbf{L}$ is rectangular, we perform dimensionality reduction by projecting the data to a reduced subspace of rank d. Thus, it allows a more compact data representation of the data and cheaper distance computations, especially for the case when original feature space is high dimensional.

Given the above, any method which learns a linear projection of the data can be considered as metric learning. For instance, principal component analysis (PCA, [Pearson, 1901]) is one of the earliest methods for performing unsupervised dimensionality reduction. To reduce the dimensionality of data, PCA seeks to find a low-rank linear projection. One of the disadvantages of PCA is that when searching for the linear projection, it aims at preserving the variance of the data and does not take into account the class membership of data points. Other works address this issue by using the supervision brought in by labelled data points. Below, we give a brief overview of different supervised methods used to learn a metric.

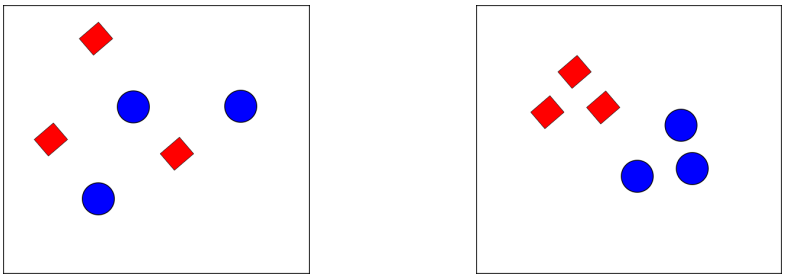


Figure 2.2 – Basic idea of metric learning is to learn a metric that assigns small distance to pairs of examples that are semantically similar. In this figure, semantically similar data points are color coded. Left: Data points in the original space, Right: Data points in the space defined by the learned metric.

Supervised methods Supervised methods differ from unsupervised methods in terms of the objective for learning the lower dimensional representation. Instead of trying to retain as much data variance as possible while projecting the data as in PCA, the goal of supervised methods is to learn a distance metric for a particular task. These methods are, by nature, supervised. Figure 2.2 shows a toy example which illustrates supervised metric learning. Many supervised Mahalanobis metric learning methods exist. Most are based on loss functions defined over pairs or triplets of data points, see *e.g.* (Davis et al. [2007], Globerson and Roweis [2006], Guillaumin et al. [2009], Köstinger et al. [2012], Mignon and Jurie [2012b], Wang et al. [2014], Weinberger and Saul [2009]).

Supervised methods for learning a distance metric can be divided into two categories: the global distance metric learning, and the local distance metric learning. The former learns a single metric shared across the input space, whereas in the latter case, multiple metrics are learned at different places in the input space. We refer the reader to recent survey papers (Bellet et al. [2013], Kulis [2012]) for a detailed review of these. Below we give a brief overview of some global and local distance metric learning methods.

2.2.2 Global Mahalanobis metric

One of the earliest supervised approaches for learning a global Mahalanobis metric is Fisher Linear Discriminant Analysis (FLDA, Fisher [1936]). Instead of searching for directions which maximize the variance of data like PCA, it searches for ones which preserve discriminating information. These directions were shown to be better than those found by PCA for the task of face recognition (Belhumeur et al. [1997]). More formally, the objective of FLDA is to find projections which maximize the between-class covariance while minimizing the

within-class covariance of the projected data. We can define the within-class covariance matrix $\mathbf{S}_W$ and between class covariance matrix $\mathbf{S}_B$ as,

$$\begin{aligned}\mathbf{S}_W &= \sum_{i \in \Omega_c} \frac{1}{\mathbf{N}_c} (\bar{\mathbf{x}}_c - \mathbf{x}_i)(\bar{\mathbf{x}}_c - \mathbf{x}_i)^\top \\ \mathbf{S}_B &= \sum_{c=1}^{C} \mathbf{N}_c (\bar{\mathbf{x}}_c - \bar{\mathbf{x}})(\bar{\mathbf{x}}_c - \bar{\mathbf{x}})^\top\end{aligned} \tag{2.3}$$

where $c \in \{1, ..., C\}$ denotes the classes, Ω_c the set of data points belonging to class c, and $\bar{\mathbf{x}_c}$ the class mean. The objective which FLDA minimizes is:

$$J(\mathbf{u}) = \frac{\mathbf{u}\mathbf{S}_B\mathbf{u}^\top}{\mathbf{u}\mathbf{S}_W\mathbf{u}^\top}. \tag{2.4}$$

The minimization of the objective in equation Eq. (2.4) leads to a generalized eigenvalue problem. The resulting projections correspond to the eigenvectors of $\mathbf{S}_W^{-1}\mathbf{S}_B$ corresponding to the largest eigenvalues.

Methods based on pairwise loss terms learn a metric so that positive pairs (*e.g.* points having the same class label) have a distance that is smaller than negative pairs (*e.g.* points with different class labels). An example of such methods is the logistic discriminant metric learning (LDML) method of Guillaumin et al. [2009]. They learn the Mahalanobis metric from pairwise supervision for the task of face verification. Using $y_{ij} \in \{-1, +1\}$ to denote whether a pair of images $\mathbf{x}_i$, $\mathbf{x}_j \in \mathbb{R}^D$ are of the same person or not, LDML seeks to learn the Mahalanobis matrix $\mathbf{M} \in \mathbb{R}^{D \times D}$ by minimizing the log-loss,

$$\mathcal{L}(\mathbf{M}, b) = \sum_{(i,j) \in C} \ln \left\{1 + \exp\left(-y_{ij}(b - d_{ij})\right)\right\}, \tag{2.5}$$

where $d_{ij} = (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j)$, and b is the learned threshold to classify the pairs being of the same person or not based on the learned Mahalanobis distance. C denotes the set of training data pairs. In (Guillaumin et al. [2010]), instead of learning the matrix $\mathbf{M}$ they propose to learn a low-rank factorization so as to explicitly regularize the learned metric to a low rank.

Davis et al. [2007] proposed "information theoretic metric learning" (ITML), where they take an information theoretic approach to learn a Mahalanobis metric. The prior knowledge is used to keep the learned metric $\mathbf{M}$ close to a known prior $\mathbf{M_0}$, which is a PSD matrix. In their work they set $\mathbf{M_0}$ to $\mathbf{I}$ (the identity matrix) and thus regularize the learned metric to stay close to the Euclidean distance metric. They interpret the similarity between the two PSD matrices via KL

divergence between Gaussian distributions which have $\mathbf{M}$ and $\mathbf{M_0}$ as covariance matrix respectively. The formulation of ITML is as follows:

$$\begin{aligned} \min_{\mathbf{M}\in\mathbb{R}^{D\times D}} \quad & KL(p(\mathbf{x};\mathbf{M}) \parallel p(\mathbf{x};\mathbf{M_0})) \\ \text{s.t.} \quad & d_{ij} \leq u \qquad \forall (i,j) \in \mathcal{S} \\ & d_{ij} \geq v \qquad \forall (i,j) \in \mathcal{D}, \end{aligned} \tag{2.6}$$

where $u, v \in \mathbb{R}$ are the threshold parameters, $\mathcal{S}$ and $\mathcal{D}$ denote the set of similar and dissimilar pairs, $p(\mathbf{x};\mathbf{M})$ is the probability density function of a Gaussian whose covariance is parametrised by $\mathbf{M}$. ITML aims at finding a metric that minimizes the sum of loss terms that enforce positive pairs to have distance smaller than a constant u and negative pairs to have a distance larger than v. One shortcoming of the method is that the matrix $\mathbf{M_0}$, needs to be picked by hand, and can have an important influence on the solution.

Köstinger et al. [2012] formulate the probability of a pair $(\mathbf{x}_i, \mathbf{x}_j)$ being similar or dissimilar via standard log-likelihood ratio test:

$$\delta(\mathbf{x}_i, \mathbf{x}_j) = \log\Big(\frac{p(\mathbf{x}_i, \mathbf{x}_j)|y_{ij} = +1}{p(\mathbf{x}_i, \mathbf{x}_j)|y_{ij} = -1}\Big). \tag{2.7}$$

Here, $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^D$ denote a pair of feature vectors, and $y_{ij} = +1$ indicates that the pair $(\mathbf{x}_i, \mathbf{x}_j)$ belongs to the same class. They obtain a Mahalanobis metric which reflects the properties of the log-likelihood ratio test. The Mahalanobis metric is defined by,

$$\mathbf{M} = \mathbf{C}_+^{-1} - \mathbf{C}_-^{-1}, \tag{2.8}$$

based on the covariance matrices $\mathbf{C}_+$ and $\mathbf{C}_-$ of positive and negative pairs. For positive pairs $\mathbf{C}_+$ is defined as $\mathbf{C}_+ = \sum_{y_{ij}=+1}(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^T$. The matrix $\mathbf{C}_-$ is analogously defined. The advantage of their method is that the solution is non-iterative and the metric can be computed in closed form. They ensure that the resulting solution is a valid metric by projecting $\mathbf{M}$ onto the cone of PSD matrices.

An example of a triplet-based approach is the large-margin nearest neighbour (LMNN) method (Weinberger et al. [2006]). This method was formulated to improve the performance of k-nearest neighbour classification. For each training instance, a neighbourhood perimeter is established. The perimeter surrounds the k nearest neighbours of the same class (targets), plus a margin. Data points with

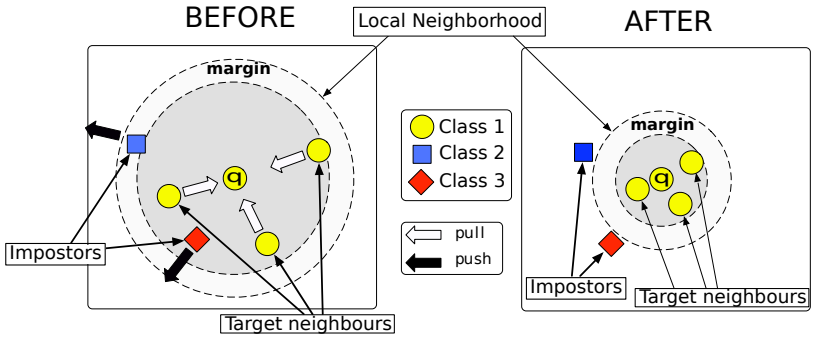


Figure 2.3 – Schematic illustration of LMNN, (left) original feature space where the impostors are equally close to the query data point q. On right, the data points in the projected feature space, learned by LMNN. The points of the target class are brought closer to query, and the impostors are pushed away by a certain margin. Image adapted from Weinberger and Saul [2009]

.

a dissimilar label which invade the perimeter are impostors. The Mahalanobis metric is learned by minimizing a sum of loss terms over triplets of points, where each loss term encourages the distance between $\mathbf{x}_i$ and its target data points to be at least one distance unit (margin) smaller than the distance of $\mathbf{x}_i$ to the impostors. Doing so, brings the target data points close to $\mathbf{x}_i$ while pushing away the impostors in the learned space. See Figure 2.3 for a graphical illustration. The objective function of LMNN is expressed as,

$$\begin{aligned} \min_{\mathbf{M}\in\mathbb{R}^{D\times D}} \quad & \sum_{(i,j)\in T} d_{ij} \\ \text{s.t.} \quad & d_{ik} - d_{ij} \geq 1 \qquad \forall (i,j) \in \mathbf{T}, \forall (i,k) \in \mathbf{I}. \end{aligned} \tag{2.9}$$

where $\mathbf{T}$ and $\mathbf{I}$ denote the set of target and impostor neighbours of the data point i, and $d_{ij} = (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{M}(\mathbf{x}_i - \mathbf{x}_j)$. They add slack variables to get soft constraints, and optimize the convex problem using a special purpose solver based on subgradient descent and careful book-keeping of active constraints. Usually the set of target neighbours is chosen and fixed using the ℓ_2 metric in the original space. This can be problematic since the ℓ_2 distance might be not be relevant in the original space. There are some recent works which address this limitation by avoiding the use of pre-defined neighbours (Weinberger and Saul [2009], Checkik et al. [2010], Trivedi et al. [2014]).

Trivedi et al. [2014] extend LMNN by optimizing a loss directly corresponding to the k-nn classification error. Their method is based on the intution that

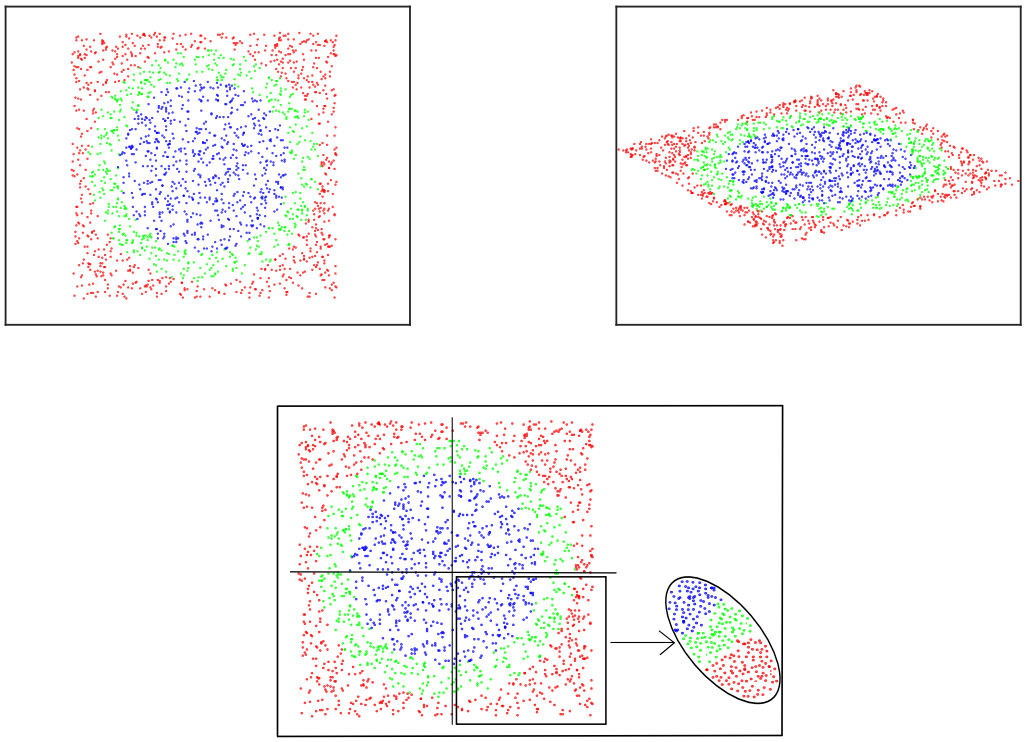


Figure 2.4 – Synthetic dataset with color coded class labels (left). Data projection given by a global Mahalanobis metric (right). Global Mahalanobis metric fails to separate the three classes given the non-linear nature of the class decision boundaries. Projection of one of the clusters of the partitioning given by a linear metric (center). This figure illustrates how locally linear metrics can separate non-linear decision boundaries in their own partition.

k-nn classifier does not require all the k neighbours to have the same label, and consequently methods like LMNN optimize a much harder objective. Unlike LMNN, which tries to push all bad neighbours (impostors) away and pull all good neighbours (targets), minimizing their proposed loss tries to push a few bad neighbours away from the given point $\mathbf{x}_i$ and pull some good neighbours, such that the resulting k-nn prediction will become correct. In contrast to many other algorithms which use pre-defined neighbours, the choice of neighbours is a latent variable which is estimated along with the distance metric.

2.2.3 Limitiations of Global Mahalanobis metric

As we mentioned before, Mahalanobis metric learning is equivalent to learning a linear projection. A linear projection on a feature space, can only implement affine transformations and might not be well suited for settings which

entail multi-modality or non-linearities in the data. We illustrate this weakness with an example. In Figure 2.4 (left panel), a synthetic toy dataset is constructed in such a way that a global linear projection cannot bring all point of a class close together, while keeping the points of different classes apart. This is observed in the data projected by using a Global Mahalanobis metric (right panel). There are different routes by which one can learn projections to separate the classes. These projections are non-linear in nature. Below, we discuss few ways to learn non-linear projections:

1. **Kernel methods** These are a class of algorithms used for machine learning or pattern analysis which operate in implicit high-dimensional spaces. Any linear model can be converted into a non-linear model, by applying the kernel trick to the model *i.e*. replacing the input features by a kernel function. Kernel trick applied with PCA, gives us kernel PCA (Schölkopf et al. [1998]). In the example of Figure 2.4, Kernel PCA with a polynomial kernel of degree two or a Guassian RBF kernel can be used to project and separate the data points.
2. **Non-linear projections** There are many different methods which can be used to learn non-linear mappings. One of these is neural networks, which consists of a series of linear projections, each followed by a non-linear activation function. The final output embedding is non-linear, where the degree of non-linearity depends upon the depth of the network and activation function used (Montufar et al. [2014]).
3. **Locally linear mappings** When the data is complex (*i.e*. multimodal or heterogenous), it might be better to learn multiple metrics at different places in the input space. For example, consider the illustration in Figure 2.4 (center), where, within each partition, learning a linear projection is able to bring the points of the same class closer while keeping the points of different classes away from each other.

The work in this chapter falls in the category of locally linear projections. Below we give a brief overview of different local Mahalanobis metric learning methods. These methods learn the projections by means of locally linear mappings.

2.2.4 Local Mahalanobis metrics

Local Mahalanobis metric learning has been shown to significantly outperform global methods on some problems, but typically comes at the expense of higher computational cost and memory requirements. Furthermore, they usually do not give rise to a consistent global metric, although some recent work

partially addresses this issue (Zhan et al. [2009b], Hauberg et al. [2012], Shi et al. [2014]). To alleviate the limitations of global Mahalanobis metric learning, many local metric learning methods have been proposed, see *e.g.* (Bhattarai et al. [2014], Bohné et al. [2014], Frome et al. [2007], Hong et al. [2011], Huang et al. [2013], Noh et al. [2010], Shi et al. [2014], Wang et al. [2012], Weinberger and Saul [2009], Zhan et al. [2009a]). Most of these are based on clustering, and learn a local metric associated with each cluster. Here we limit our discussion to recent state-of-the-art methods relevant to our work.

Weinberger and Saul [2009] extend their previous work (Weinberger et al. [2006]) to learn local Mahalanobis metrics. The data points are grouped in clusters in accordance with their class labels, following which, they learn a Mahalanobis metric per cluster and optimize the same objective as LMNN. To compare a test data point $\mathbf{x}_i$ with data point $\mathbf{x}_j$ in the training set, they use the metric associated with the cluster of $\mathbf{x}_j$. More formally the Mahalanobis distance is expressed as,

$$d(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x_i} - \mathbf{x}_j)^\top \mathbf{M}^{y_j} (\mathbf{x_i} - \mathbf{x}_j), \tag{2.10}$$

where $\mathbf{M}^{y_j}$ is metric associated to the cluster of $\mathbf{x}_j$, indexed by its label y_j. Note, that the resulting metric is not symmetric w.r.t. to its input arguments and hence is not a valid Mahalanobis metric.

Wang et al. [2012] propose a parametric local metric learning alogrithm, where the matrix function varies smoothly over the data manifold. They parametrize the distance metric for an instance as a weighted combination of local metrics, which are defined over a small set of anchor points. The anchor points are given by the means of the clusters constructed with k-means clustering. The proposed algorithm consists of two steps, where in the first step for each data point they learn a set of weights by approximating the data point as a linear combination of anchor points. They minimize the reconstruction error and incorporate manifold regularization in the objective so as to ensure that the weights vary smoothly over the manifold.

In the second step, they fix the weights and learn the local metrics using the LMNN objective function. The final distance metric to compute distances from $\mathbf{x}$ to other points is given by a weighted sum of the metrics, using the weights for $\mathbf{x}$ over the metrics. The distance metric is expressed as,

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_s \mathbf{W}_{is} (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{M}_s (\mathbf{x}_i - \mathbf{x}_j), \tag{2.11}$$

where $\mathbf{W}_{is} \in \mathbb{R}$ is the weight of the local metric $\mathbf{M}_s$ for the data point $\mathbf{x}_i$. Like LMNN-local, the distance metric they propose is not symmetric, and hence not a

valid metric. At test time, to set the weights for the test data points, they simply set them as the weights of the nearest neighbour training point. This lookup is costly when large training sets are used in practice.

The R^{2}LML method (Huang et al. [2013]) jointly learns a set of local metrics $\mathbf{M}_s$ and weights g_i^s, that assign data points $\mathbf{x}_i$ to the local metrics indexed by s. The distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ is computed using the weighted sum of metrics, where metric s is weighted by the product $g_i^s g_j^s$. The resulting metric for comparing data points $\mathbf{x}_i$ and $\mathbf{x}_j$ is expressed as,

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_s g_i^s g_j^s (\mathbf{x}_i - \mathbf{x}_j)^\top \mathbf{M}_s (\mathbf{x}_i - \mathbf{x}_j). \tag{2.12}$$

The final metric is symmetric w.r.t. to its input and hence is a valid metric. They iteratively learn the weights and the metrics, updating one while keeping the other fixed. Minimizing their objective encourages the similar points to be close to each other, while encouraging the separation between dissimilar points to be larger than 1. To ensure low-rank matrices, they use nuclear norm as a regularizer. To determine the weights over the metrics for test points, the weights of the nearest training point are used, which again implies a costly lookup when large training sets are used in practice.

Shi et al. [2014] make the following observation: any Mahalanobis matrix $\mathbf{M} \in \mathbb{R}^{D \times D}$ can be expressed as a conic combination of rank-1 PSD matrices:

$$\mathbf{M} = \sum_{s=1}^{k} w_s \mathbf{b}_s \mathbf{b}_s^\top, \qquad \forall w_s \geq 0, \tag{2.13}$$

where $\mathbf{b}_s$ are D dimensional column vectors. For the case of PCA, $\mathbf{M}$ is obtained by setting $\mathbf{b}_s$'s as the leading eigen vectors, and w_s with their corresponding eigenvalues; the eigenvectors are given by eigendecomposition of the data covariance matrix. In their paper they cast the problem of metric learning as learning sparse combinations of a large base set of rank-1 base metrics. The rank-1 base metrics are found by clustering the dataset, and applying Fisher linear discriminant analysis (FLDA) in each cluster. They concatenate the rank-1 base metrics found in different clusters to form a set, and use 400 elements from the set to form a basis. Given a basis of k rank-1 matrices, learning a global metric involves learning k parameters compared to D^2 as in most metric learning algorithms. They learn the metric by optimizing the LMNN objective along with a ℓ_1 norm regularization to encourage sparse solutions. They extend their work for local metric learning by taking a similar approach as (Noh et al. [2010],

Wang et al. [2012]), and measure the distance between a test point $\mathbf{x}$ and a training point $\mathbf{x}_i$ by using a weighted combination of base metrics. The metric tensor for the point $\mathbf{x}_i$ is expressed as,

$$\mathbf{M}(\mathbf{x}_i) = \sum_{s=1}^{k} \phi^s(\mathbf{x}_i)\mathbf{b}_s\mathbf{b}_s^\top. \tag{2.14}$$

Here $\phi^s(\mathbf{x}_i)$ is the weight for the s^{th} basis, which parametrically depends on embedding of $\mathbf{x}_i$:

$$\phi^s(\mathbf{x}_i) = (\mathbf{a_s}^\top \mathbf{z_i} + c_s)^2, \tag{2.15}$$

here $\mathbf{z_i} \in \mathbb{R}^{d'}$ is an embedding of $\mathbf{x}_i$ obtained with Kernel-PCA, $\mathbf{A} = [\mathbf{a}_1 ... \mathbf{a}_k]$ is a $d' \times k$ matrix and $c_s \in \mathbb{R}^k$. Learning $\mathbf{A}$ and c, they implicitly learn the weight of the base metrics for the training data but also for any point in the input feature space. The total number of learnable parameters are $k(d' + 1)$. The advantage of their approach is that weights are easily evaluated for new test points. Also the total number of learnable parameters are independent w.r.t. the number of data points and the input dimensionality of data. A limitation, however, is that a fixed set of base metrics given by FLDA restricts the class of metrics that can be learned. This is particularly detrimental for high-dimensional data.

Bohné et al. [2014] proposed LMLML, an approach based on GMM clustering, which learns a metric associated with each cluster. To compare two points $\mathbf{x}_i$ and $\mathbf{x}_j$ they use a weighted sum of the local metrics, where the weight of each metric is given by $p(s|\mathbf{x}_i) + p(s|\mathbf{x}_j)$: the sum of the soft-assignments for $\mathbf{x}_i$ and $\mathbf{x}_j$ to the GMM components. If two points are far away, however, it is not clear that the local metric associated with either data point will be appropriate for a pair-wise comparison. Therefore, they also add a learned global metric to the weighted sum of metrics. The final resulting metric tensor is given by,

$$\mathbf{M}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{M}_0 + \sum_{s=1}^{k} w_s(\mathbf{x}_i, \mathbf{x}_j)\mathbf{M}_s, \tag{2.16}$$

where $w_s(\mathbf{x}_i, \mathbf{x}_j) = p(s|\mathbf{x}_i) + p(s|\mathbf{x}_j)$, and $\mathbf{M}_0$ is the global metric.

Bhattarai et al. [2014] proposed a hierarchical method for efficient retrieval that learns a hierarchical clustering of the data by interleaving metric learning and k-means clustering. More formally, they start by taking all the data points present in the root node and learn a discriminative subspace with the Mahalanobis metric. They optimize an objective involving pairwise constraints, similar to LDML. Once the metric is learned, they project the data in the subspace

and perform k-means clustering to obtain two child nodes. They follow this procedure recursively, upto a pre-defined depth. At the end, each element in the training set is assigned to a leaf of the hierarchy based on the local metrics and clustering. A query is assigned to a leaf node, and retrieval is performed among the data in that leaf-node, using the associated metric. Their hierarchical decomposition speeds up the retrieval since only a fraction of the dataset is accessed for a given query. They report improved retrieval accuracy due to the use of local metrics, as compared to the one obtained with the use of global metric.

None of these methods allow the local metrics to be expressed as the ℓ_2 distance after a non-linear data transformation. The work of (Hauberg et al. [2012]) is an exception in this respect: they show that if local metrics vary smoothly in the input space, then they form a Riemannian metric on the data manifold. They define a smoothly varying local metric as a linear combination of a fixed set of local metrics, which are learned separately using any local metric learning algorithm. They perform PCA in the Riemannian metric to obtain a global Euclidean data representation. They show their framework improves w.r.t. Euclidean PCA.

Limitations Except for the work of Hauberg et al. [2012], none of these methods allow the local metrics to be expressed as the ℓ_2 distance after a non-linear data transformation. This means that the local metrics cannot be used for global data visualization, and do not support efficient retrieval techniques based on ℓ_2 quantization, such as product quantization and multiple assignment retrieval (Jégou et al. [2011]).

2.3 Globally aligning local Mahalanobis metrics

As mentioned before, the majority of work on local metrics does not allow the local metrics to be expressed as ℓ_2 distance after the projection. In other words, data points once projected, lack a common distance measure between all data points. For example in (Weinberger and Saul [2009], Bhattarai et al. [2014]), the distance measure is valid within a region, whereas for (Shi et al. [2014], Bohné et al. [2014], Wang et al. [2012], Huang et al. [2013]) the distance measure varies per data point. In our work, we address this issue by learning local metrics and embedding them in a global low-dimensional representation, in which the ℓ_2 metric can be used.

To obtain a more general class of metrics, we define several local Mahalanobis metrics. We cluster the data using a k-component Gaussian mixture

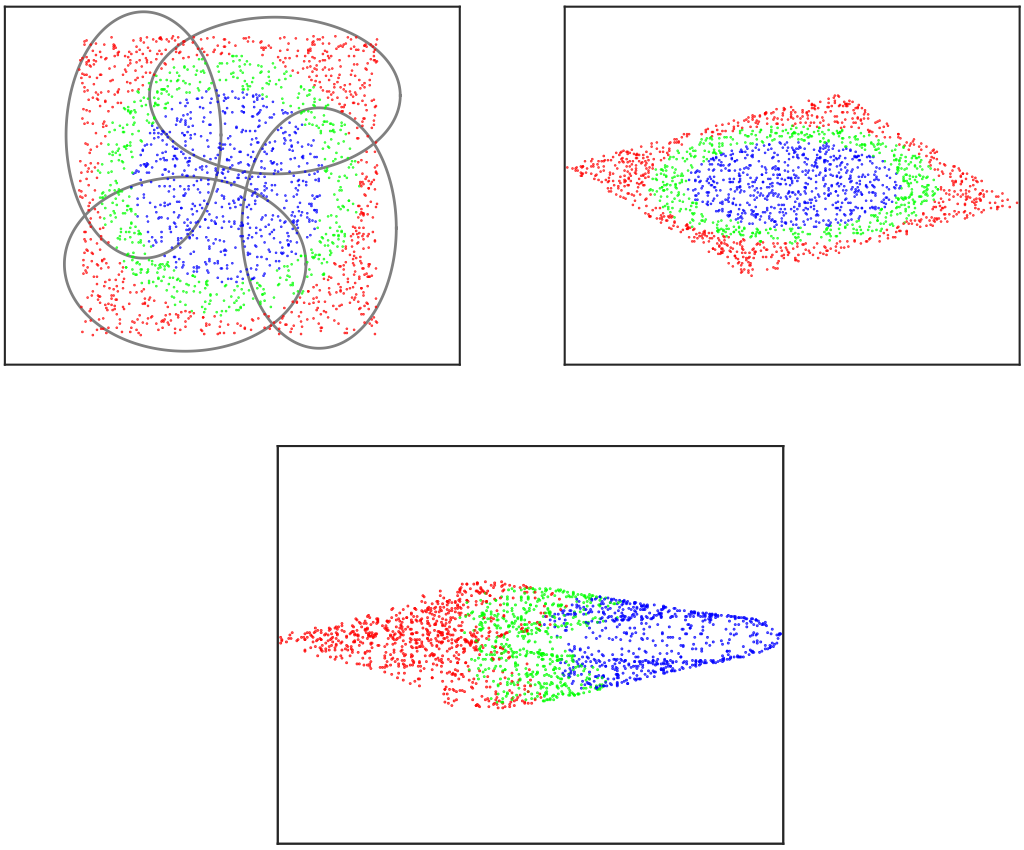


Figure 2.5 – Synthetic dataset with color coded class labels, and the GMM used by our CLML local metric (left panel). Data projection given by a global Mahalanobis metric (right panel) and our local CLML metric (bottom panel). The latter approximately separates the classes via local linear maps.

model (GMM),

$$p(\mathbf{x}) = \sum_{s=1}^{k} p(s)p(\mathbf{x}|s) = \sum_{s=1}^{k} \pi_s \mathcal{N}\left(\mathbf{x}; \mu_s, \Sigma_s\right), \tag{2.17}$$

where π_s, μ_s, and Σ_s are respectively the mixing weight, mean, and covariance matrix of cluster s. The posterior distribution $p(s|\mathbf{x}) = p(s)p(\mathbf{x}|s)/p(\mathbf{x})$ defines a soft-assignment of the data over the k clusters.

We can compute distances between points assigned to the same cluster s using a local metric learned for that cluster, defined by a local projection matrix $\mathbf{L}_s$. It is, however, not clear how to compare vectors that are assigned to different clusters. In order to combine the local metrics, we define a global representation

in which we integrate the local projections given by the different $\mathbf{L}_s$. Similar to global Mahalanobis metric learning, and unlike previous work, our formulation amounts to projecting the data (in a locally linear way) to a new representation, and computing the ℓ_2 metric in this new representation. This allows us to compute distances between any pair of samples, regardless of their cluster assignments, hence allowing us to visualize all data in a single view, and the use of ℓ_2-based efficient retrieval techniques. In Figure 2.5, as a proof of principle, we illustrate the performance of our "coordinated local metric learning" (CLML) approach on the toy dataset. Using CLML we obtain a data representation that respects the pairwise training constraints much better (bottom panel) compared to the representation found by a global metric (right panel).

2.3.1 Coordinated local metric learning

As pointed out above, we can interpret a Mahalanobis metric as computing the ℓ_2 distance after linear projection of the data. Local Mahalanobis metrics can therefore be interpreted as locally mapping the data points $\mathbf{x}$ to several different, local, coordinate systems via projections $\mathbf{L}_s\mathbf{x}$. The projection of data point $\mathbf{x}_i$ for a local coordinate system s is expressed as,

$$\mathbf{z}_{is} = \mathbf{L}_s\mathbf{x}_i. \tag{2.18}$$

Given these local coordinate systems, our goal is to align them across different local models. Since the ℓ_2 metric is invariant to rotations, translations and reflections, we can use them to arbitrarily modify the local projection $\mathbf{z}_{is}$ to,

$$\mathbf{z}_{is} := \mathbf{R}_s\mathbf{L}_s\mathbf{x}_i + \mathbf{b}_s, \tag{2.19}$$

where $\mathbf{R}_s$ denotes an orthonormal matrix, *i.e.* for which $\mathbf{R}_s^\top\mathbf{R}_s = \mathbf{I}$, which can implement rotations and reflections, and $\mathbf{b}_s$ denotes a translation vector. It is easy to verify that

$$\| \mathbf{z}_{is} - \mathbf{z}_{js} \|_2^2 = \| \mathbf{L}_s\mathbf{x}_i - \mathbf{L}_s\mathbf{x}_j \|_2^2 . \tag{2.20}$$

In this way, we can learn the local metrics $\mathbf{L}_s$ and their alignment parameters to obatin a globally consistent embedding. In particular, given that $\mathbf{x}_i$ and $\mathbf{x}_j$ are assigned to different clusters r and t respectively, we can set the $\{\mathbf{R}_s, \mathbf{b}_s\}$ to ensure that $\mathbf{z}_{ir}$ and $\mathbf{z}_{jt}$ are close if $\mathbf{x}_i$ and $\mathbf{x}_j$ form a positive pair, and far away if they form a negative pair.

Instead of learning the $\{\mathbf{R}_s, \mathbf{b}_s\}$ for fixed $\mathbf{L}_s$ that were learned in advance, we will learn both the local metrics and their alignment in a joint manner. To that

end, we can absorb $\mathbf{R}_s$ into $\mathbf{L}_s$ without loss of generality, and define a mapping of the data points $\mathbf{x}_i$ to a global coordinate system as

$$\mathbf{z}_i := \sum_{s=1}^{k} q_{is}\mathbf{z}_{is} = \sum_{s=1}^{k} q_{is}\big(\mathbf{L}_s\mathbf{x}_i + \mathbf{b}_s\big), \tag{2.21}$$

where $q_{is} := p(s|\mathbf{x}_i)$ is the soft-assignment of $\mathbf{x}_i$ to cluster s. In the case of hard-assignments (*e.g.* k-means), $\mathbf{z}_i$ is given by the local projection of the cluster to which it is assigned. In the case of soft-assignments (*e.g.* GMM), $\mathbf{z}_i$ is the weighted average of the local projections to the global representation. Note that in the global coordinates given by the $\mathbf{z}_i$ we can compare any pair of points, regardless of whether they are assigned to different clusters or not. Our goal is now to learn $\{\mathbf{L}_s, \mathbf{b}_s\}$ so that $\mathbf{z}_i$ and $\mathbf{z}_j$ are close for positive pairs, and far away for negative pairs. We assume the GMM clustering is fixed.

In (2.21), we can see that $\mathbf{z}_i$ is dervied as a weighted sum of locally linear projections. Therefore, we can re-write this weighted sum as a single linear projection

$$\mathbf{z}_i = \tilde{\mathbf{L}}\phi(\mathbf{x}_i), \tag{2.22}$$

where $\tilde{\mathbf{L}} = (\mathbf{L}_1, \mathbf{b}_1, \ldots, \mathbf{L}_k, \mathbf{b}_k)$ collects the local linear projections, and the augmented data representation $\phi(\mathbf{x}_i) = \big(q_{i1}(\mathbf{x}_i^\top, 1), \ldots, q_{ik}(\mathbf{x}_i^\top, 1)\big)^\top$ containing k copies of $\mathbf{x}_i$ appended with a one, each weighted by the corresponding soft-assignment. The projection matrix $\tilde{\mathbf{L}}$ defines the local metrics used to compare points that are assigned to the same cluster, but also the rotations, reflections, and translations to globally align the local representations. The augmented data representation depdends only upon the clustering algorithm used for clustering and can be applied to data as a pre-processing.

For a given partitioning of the data, the $\mathbf{z}_i$ are obtained as a linear projection of the transformed input vectors $\phi(\mathbf{x}_i)$, the ℓ_2 distance between $\mathbf{z}_i$ and $\mathbf{z}_j$ is therefore equivalent to a Mahalanobis distance between $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$.

$$\| \mathbf{z}_i - \mathbf{z_j} \|_2^2 = (\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j))^\top \tilde{\mathbf{M}}(\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)), \tag{2.23}$$

with $\tilde{\mathbf{M}} = \tilde{\mathbf{L}}^\top\tilde{\mathbf{L}}$. Therefore, the problem of learning a globally aligned ensemble of local metrics takes the same form of learning a global Mahalanobis metric; be it using the expanded high-dimensional data representation given by the $\phi(\mathbf{x}_i)$. As a result, existing Mahalanobis metric learning methods can be used to learn the projection matrix $\tilde{\mathbf{L}}$ for CLML.

2.3.2 Discussion

It is easy to see that CLML generalizes Mahalanobis metrics. For $k \geq 1$, if each cluster s uses the same projection given by $\mathbf{L}_s = \mathbf{L}$ and $\mathbf{b}_s = \mathbf{b}$, then for arbitrary soft-assignments $\mathbf{z}_i = \mathbf{L}\mathbf{x}_i + \mathbf{b}$ is a linear projection of $\mathbf{x}_i$, and the ℓ_2 distance between $\mathbf{z}_i$ and $\mathbf{z}_j$ is a Mahalanobis distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ given by $\| \mathbf{L}(\mathbf{x}_i - \mathbf{x}_j) \|_2$. With proper regularization, we therefore expect performance that is at least on par with global metric learning.

Our work is also related to the local linear manifold learning technique of Teh and Roweis (Teh and Roweis [2003]). They use a mixture of factor analyzers (MFA) (Ghahramani and Hinton [1996]) to map data points to local low dimensional coordinate systems associated with the mixture components. To align the local coordinates, they minimize the Locally Linear Embedding (LLE) (Roweis and Saul [2000]) objective function. Our work differs in that we learn coordinated local linear projections in a supervised manner. Also, in our work we use a diagonal covariance GMMs —which are faster to train than MFA— and learn local linear maps directly from the original feature space to the global representation instead of mapping from the local MFA subspaces. Since the MFA is learned by optimizing a different unsupervised cost function, the obtained subspaces might be suboptimal.

Our work is similar to Hauberg et al. [2012] in the sense that in both cases the local metrics can be expresses as a ℓ_2 distance after a linear transformation. Also, in line with Hauberg et al. [2012], our metric tensor uses squared exponentials in weight functions, and forms a Riemannian metric on the data manifold. Our approach differs in that, (i) we learn the local metrics and their alignment in a joint manner, and (ii) to project a point to the global representation Hauberg et al. [2012] needs to solve a system of second-order differential equations with size quadratic with the data dimension, whereas our approach requires only averaging local linear projections.

The expanded data representation $\phi(\mathbf{x}_i)$ can be seen as an explicit feature map embedding of a kernel, which in this case is the Fisher kernel [Jaakkola and Haussler, 1999]. Indeed, we can see that $\phi(\mathbf{x}_i)$ is a special case of Fisher vector image representation for GMMs [Sánchez et al., 2013], when considering just the Fisher vector components corresponding to the Gaussian means. Learning a linear projection on the explicit feature embedding can be equivalently done by learning the Mahalanobis distance in the kernel space, with the kernel function

defined as,

$$k(\mathbf{x}_i, \mathbf{x}_j) = \Big(\sum_{s=1}^{k} q_{is} q_{js}\Big)\Big(\mathbf{x}_i^\top \mathbf{x}_j + 1\Big) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j). \tag{2.24}$$

The kernel function is a product of two kernels. One kernel is linear in the input data, and the other kernel is non-linear, with the non-linearity introduced by clustering. The explicit embedding of the kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ is given by the vectorized outer product of the two kernels, which in our case is equal to the expanded date representation $\phi(\mathbf{x}_i)$.

2.3.3 Implementation

Optimization We use the LDML (Guillaumin et al. [2009]) objective function to learn our local metrics parameterized by $\tilde{\mathbf{L}}$. Let the label $y_{ij} \in \{-1, +1\}$ denote whether $(\mathbf{x}_i, \mathbf{x}_j)$ is a positive or a negative pair. LDML then minimizes the log-loss

$$\mathcal{L}(\tilde{\mathbf{L}}, b) = \sum_{i,j} \ln\left\{1 + \exp\left(-y_{ij}(b - ||\mathbf{z}_i - \mathbf{z}_j||^2)\right)\right\}, \tag{2.25}$$

where b is a scalar (estimated along with $\tilde{\mathbf{L}}$) that determines at which distance pairs are considered positive or negative. We add a Frobenius norm regularizer over $\tilde{\mathbf{L}}$ to avoid overfitting, and cross-validate the regularization weight. We use a global LDML metric to initialize the local metrics. In our implementation we use the sum formulation of Eq. (2.21), which avoids explicitly storing the $\tilde{\mathbf{x}}_i$. Compared to global metric learning, we only need to additionally store the soft-assignments. In practice this is a negligible overhead, in addition the assignments can be thresholded to be sparse. The cost to compute the z_i increases sub-linearly with k because of this sparsity. For example for $k = 32$ clusters we typically get five soft assignments larger than 10^{-3}.

Clustering To partition the input space in CLML, we learn diagonal covariance GMMs. In our experiments we consider two alternatives for the data on which the GMMs are learned. We either learn the mixture in the original feature space, or learn the mixture in the projection space obtained by global LDML metric learning. The rationale for the latter option is that the GMM clustering will be more meaningful in the global metric learning space.

Efficient retrieval For efficient face retrieval we use the multiple assignment approach of (Jégou et al. [2011]). The $\mathbf{z}_i$ in the retrieval set are clustered using

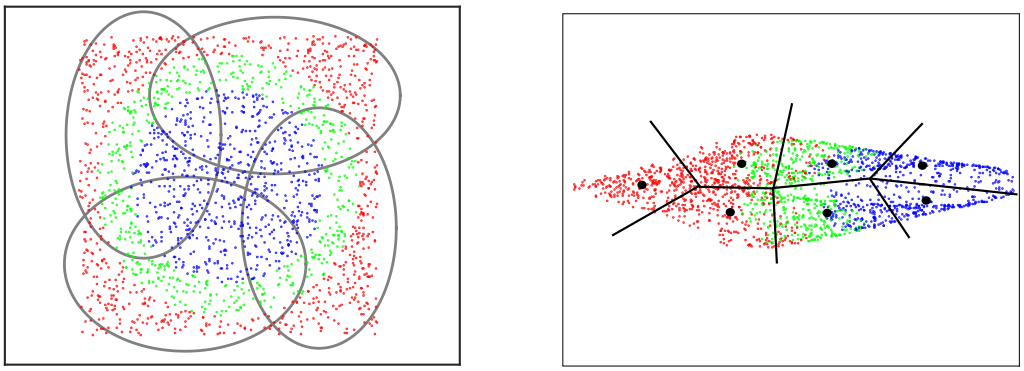


Figure 2.6 – Illustration of quantization based retrieval method used in conjunction with CLML. On the left, the input data clustered with a 4 component Gaussian Mixture Model. On the right, the data representation given by the learned CLML projection clustered using k-means for quantization based retrieval.

k-means, and each $\mathbf{z}_i$ is assigned to the nearest center. A query $\mathbf{z}$ is assigned to the m closest k-means centers. Only points assigned to these m centers are returned, ranked by their distances to the query $\mathbf{z}$.

The representation obtained with CLML embeds the data points in a global coordinate space where all the pairwise distances are valid. Once the data is projected with CLML we use the multiple assignment approach for efficient retrieval. Doing so, allows us to decouple the number of local metrics used for CLML and the number of clusters used for multiple assignment approach. See illustration in Figure 2.6.

2.4 Experimental evaluation

In this section we describe our experimental evaluation of the coordinated local metric learning. We validate our method for the task of face retrieval and verification. We start with providing details of the data sets, protocols and the performance measures. Then we briefly describe the features we have used to represent the face images, and finally we describe the results for image retrieval and verification.

2.4.1 Dataset, protocols and features

For our experiments we use the Labeled Faces in the Wild (LFW) (Huang et al. [2007]) dataset. It contains a total of 13,233 faces of 5,749 people collected from the web. The dataset was designed for verification experiments, where we

have to determine for a pair of face images if they depict the same person or not. The identities present in the train and test set are disjoint, so that we measure the ability to do verification on people not seen during training. In our experiments, we use the standard "unrestricted" training protocol, that allows the use of all pairs in the training set in contrast to the "restricted" protocol which only allows the use of 600 face pairs per training fold which are labeled as matching or non-matching identities. The verification accuracy is measured using ten-fold cross-validation. The train set is used to learn a metric, and to estimate a threshold on the metric. Using these, the pairs in the test set are classified as positive or negative, and the accuracy of this classification is reported.

Since the LFW verification accuracy is saturating in recent years (Liao et al. [2014]), we focus on the more challenging retrieval-based evaluation of Bhattarai et al. [2014]. The set of 423 queries consists of one image of each person in LFW with five or more images. All images not in the query set form the retrieval set, and are used to learn the metric. We augment the retrieval set with up to one million distractor faces provided by Bhattarai et al. [2014], which belong to people not present in the LFW dataset. The 1-call@n performance measure is the fraction of queries for which at least one of the top n ranked result faces is of the same person. They report the 1-call@n for a range of values of n, giving a full 1-call@n curve. We also use the mean average precision (mAP) measure, which gives us a single number per setting instead of a full 1-call@n curve.

Image pre-processing and features We align face images with a similarity transform based on detection of points on the eyes, nose, and mouth, see Everingham et al. [2009]. We consider three different representations. The first is the LBP features of Bhattarai et al. [2014], which allows direct comparison to their work. We compute a 9,860 dimensional descriptor by concatenating 58 dimensional LBPs (Ojala et al. [2002]) on each cell of a 10×17 grid over the face. The second is similar to the Fisher vector (FV) features of Simonyan et al. [2013a]. We densely compute at each pixel a root-SIFT descriptor (Arandjelovic and Zisserman [2012]), using 24×24 pixel patches. The descriptors are projected to 64 dimensions using PCA. Spatial layout information is incorporated by appending the 2D image coordinates of the patch center to the descriptors (Sánchez et al. [2012]), providing a 66 dimensional local feature. We represent the face by computing a 16,896 dimensional FV (Sánchez et al. [2013]) using 128 Gaussian components. The third feature is derived from the penultimate layer of a convolutional neural network trained on the CASIA WebFace dataset (Yi et al. [2014]), which contains 494,414 faces of 10,575 subjects. The network architecture is similar to the one proposed in Yi et al. [2014] and the dimen-

	GMM feature space	
Nr. of local metrics k	Original	Global metric
2	73.04	73.09
4	74.04	75.18
6	73.58	75.24
8	73.92	75.59

Table 2.1 – Evaluation of CLML over FV features using different GMM clustering methods. All learned metrics project the data to $d = 32$ dimensions. Performance is measured in mAP, higher is better.

sionality of the extracted feature is 320. We differ slightly from the architecture proposed in Yi et al. [2014]: (i) We optimize the classification loss, instead of optimizing both classification and verification losses. (ii) We do not use dropout for regularization.

2.4.2 Experiments for retrieval

Here we present the experimental results for the task of face retrieval, where we systematically explore the effect of different choices and parameters on the performance of CLML. For comparability with Bhattarai et al. [2014], we use projections to $d = 32$ dimensions unless stated otherwise. Finally, we also present the results for large-scale retrieval setting, and show the efficiency of quantization based methods for efficient retrieval when used in conjunction with CLML.

Comparison of CLML with global metric learning In Table 2.1 we compare CLML using GMMs trained either in the original FV features, or on data projected to $d = 32$ dimensions by a global LDML metric, *c.f.* Section 2.3.3. In all cases, the clustering obtained LDML projections leads to better results. Where the difference between the two clustering approaches is only 0.05 for $k = 2$ local metrics, it increases for larger numbers of clusters, up to 1.67 for $k = 8$ local metrics. In all subsequent experiments we therefore use clustering using LDML projections. Clustering in the LDML projected space is also much faster: in this case it takes about 0.5 secs. to learn the GMM on 10,000 points.

In Table 2.2 we evaluate CLML on the three features, while varying the number of local metrics. We also state results obtained when cross-validating the number of local metrics, as well as results of global LDML metrics and the ℓ_2

	Features		
Nr. local metrics k	LBP	FV	CNN
2	41.51	73.09	59.78
4	44.02	75.18	61.43
8	47.94	75.59	64.64
16	49.00	76.20	66.07
32	49.98	75.61	70.98
64	49.70	75.58	73.83
Cross-validated	49.89 (26)	74.99 (28)	73.83 (64)
Global LDML metric	36.95	68.12	58.46
ℓ_2 metric	13.24	22.88	63.06

Table 2.2 – Performance of CLML in retrieval mAP for the three features, while varying the number of local metrics.

metric. The results lead to the following observations. (i) CLML generally improves when using more local metrics. (ii) Cross-validation over the number of local metrics successfully selects a (near) optimal number of local metrics. In subsequent experiments we cross-validate the number of local metrics for CLML. (iii) The FV features lead to better results than the LBP and CNN features. (iv) For all tested settings, CLML consistently improves over LDML.

In Figure 2.7 we compare CLML and LDML for the three features across a range of projection dimensions. The results show that CLML consistently improves over LDML for all projection dimensions with the three features. The improvements are particularly large for the CNN and LBP features. For the LBP and FV features, the best results are obtained with CLML at $d = 128$, with $k = 16$ set by cross-validation: 61.6% and 82.2% mAP respectively. LDML with the same number of parameters, *i.e.* with $d = 128 \times 16 = 2048$, obtains 53.0% and 80.9% respectively. This shows that the improvement of CLML is not simply because it has more trainable parameters. In the case of CNN descriptors, the best performance is obtained with CLML at $d = 128$ and $k = 64$ set by cross-validation: 76.95% mAP. Since the descriptors are only 320 dimensional, we cannot compare to LDML with the same number of parameters. Using a full-rank 320×320 Mahalanobis metric, however, yields 63.71% mAP, which is significantly worse.

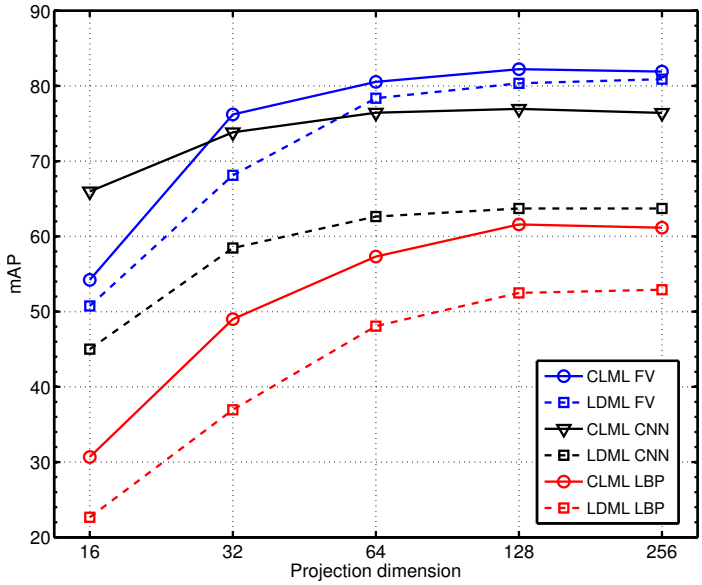


Figure 2.7 – Performance in mAP of local and global metrics for the three features, using projection dimensions from 16 to 256.

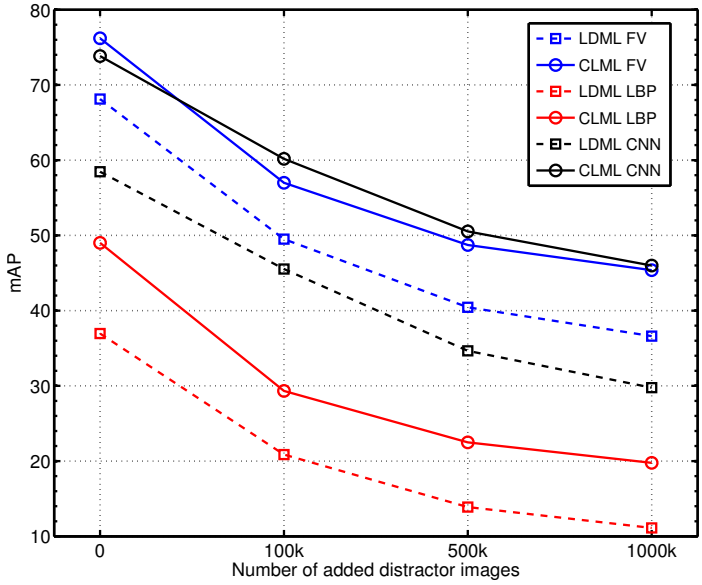


Figure 2.8 – Performance of CLML and LDML with different numbers of distractors added to the LFW images.

Large-scale face retrieval experiments In our second set of retrieval experiments we add up to one million additional distractor images to the LFW images.

In Figure 2.8 we evaluate the results of CLML and LDML for the three features, while increasing the number of distractors from zero to one million. We observe that performance degrades gracefully, and that the improvement of CLML over LDML is stable as a function of the number of distractors for all three features.

In Figure 2.9 we make a direct comparison to Bhattarai et al. [2014]: replotting the 1-call@n curves reported there. From the other state-of-the-art methods discussed in Section 2.2 we compare to SCML (Shi et al. [2014]). [1] For LMLML (Bohné et al. [2014]) code is not available, while for R^{2}LML (Huang et al. [2013]) we found the code too inefficient to use with our high dimensional features.

For [Bhattarai et al., 2014] we report the results for their "256D4" setting, which they found to give best results and uses eight local metrics, and also include their global metric learning results obtained with PCCA (Mignon and Jurie [2012b]). Our results are directly comparable, since we use the same LBP features and also learn $d = 32$ dimensional projections. Our CLML results substantially improve over the results of Bhattarai et al. [2014], *e.g.* from under 40% to over 70% 1-call@n for $n = 10$ for the case without distractors (Figure 2.9, top panel). Interesting we also obtained large improvements over Bhattarai et al. [2014] using global LDML metrics.

To understand the large performance difference, we re-implemented their approach (Figure 2.9, top panel, green curve), and obtained improvements of about 10 points w.r.t. their results. We found that most of this improvement is due to the ℓ_2 regularization that we use, but Bhattarai et al. [2014] did not. We also implemented a non-hierarchical variant of their approach, based on "flat" k-means clustering, but which is otherwise the same (Figure 2.9, top panel, black curve). This leads to another improvement of about 10 points, which suggests that flat clustering leads to clusters that are better suited for retrieval. Our global metric learning results obtained with LDML (Figure 2.9, top panel, red dashed curve) are yet another 10 points better. This shows that the benefit of using local metrics is counterbalanced by only retrieving points assigned to the same cluster as the query, as is done by Bhattarai et al. [2014] and for the green and black curves.

1. Code available at http://mloss.org/software/view/553.

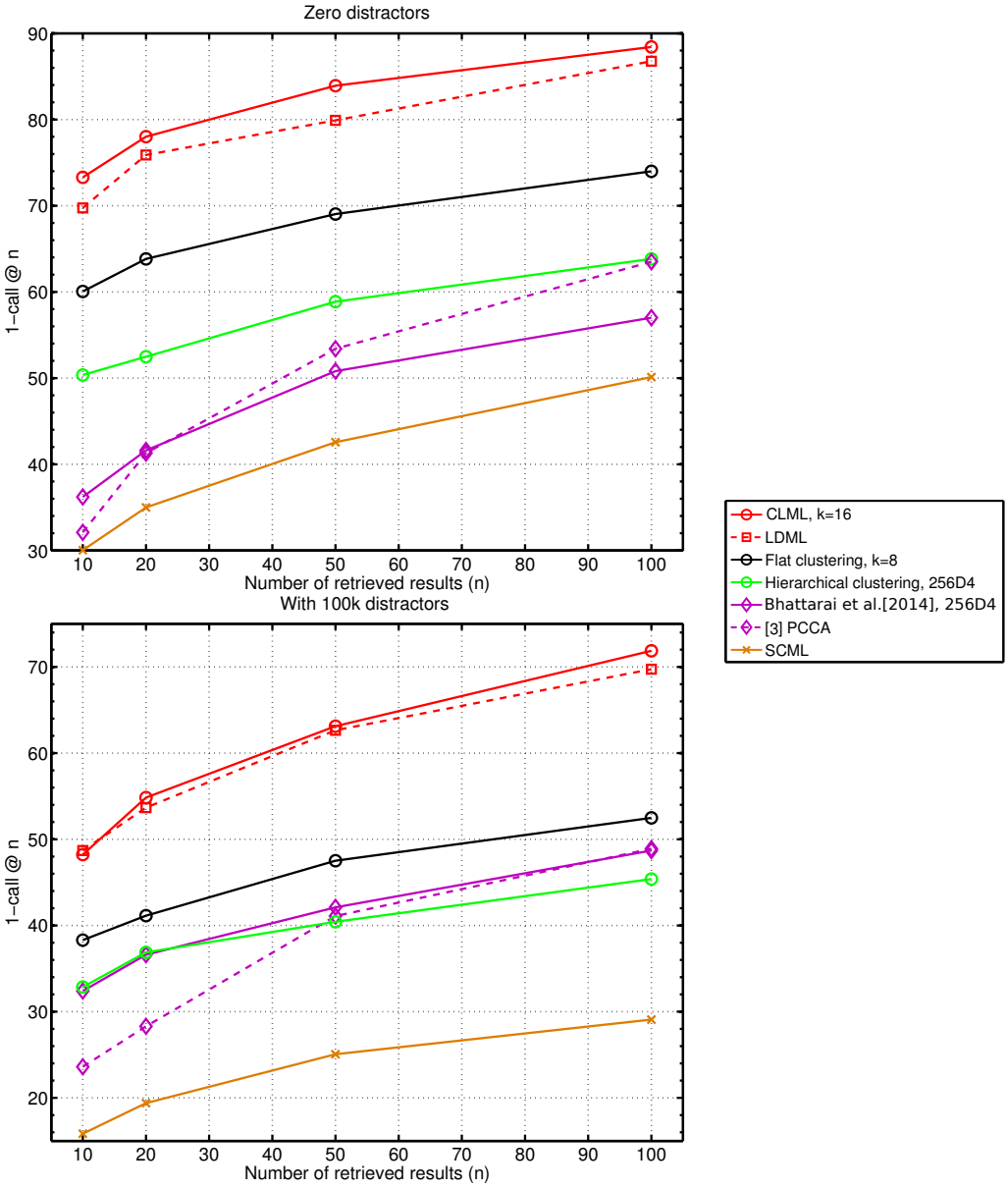


Figure 2.9 – Retrieval using LFW images only (top), and using LFW plus 100,000 distractor faces (bottom). The results marked with Bhattarai et al. [2014] correspond to those reported therein. Results for SCML and LDML have been produced using publicly available code. See text for details.

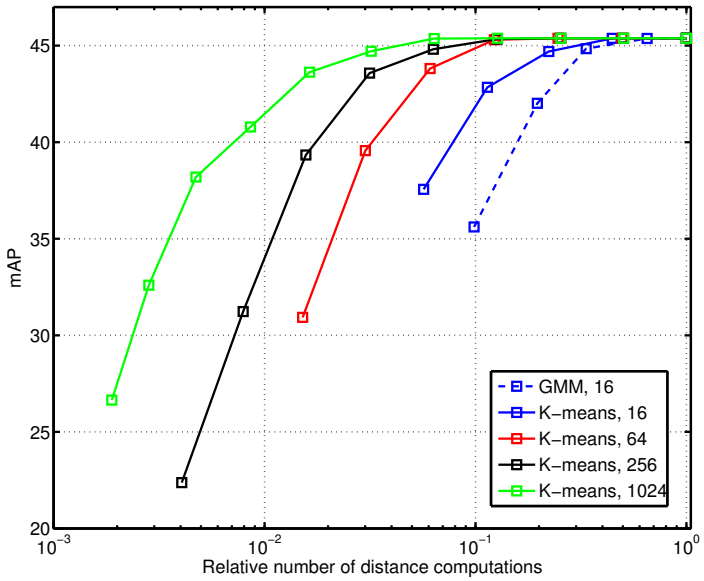


Figure 2.10 – Retrieval mAP and speed on LFW plus one million distractor faces, using CLML with $d = 32$ and FV feaures. Varying the number of quantization cells p, and number of assignments m.

In Figure 2.9 (bottom panel), we evaluate the results with 100k distractor images added to the retrieval set. From the figure we can observe that performance of all the methods drop by approximately 20 points. Similar to what we observed in the setting with zero distractors, CLML substantially improves over the results of Bhattarai et al. [2014] and other methods.

Using SCML (Shi et al. [2014]) we obtained the worst retrieval results. This is because SCML learns metrics using a limited set of base metrics, which is detrimental for high-dimensional data. To improve results we tuned the number of base metrics (600 gave best results), and also excluded faces of people with less than 3 images to compute the base metrics with FLDA, which also improved the results. The number of clusters used to produce the base metrics in SCML is another hyper-parameter that might require further tuning; for our experiments we use the default setting, where the number of clusters are set via a heuristic based on the number of base metrics and dimensionality of the input data.

Efficient retrieval with CLML metrics CLML projects all data in a single representation in which the ℓ_2 distance is used. Therefore we can decouple the clustering used for local metric learning, and the clustering used for the effi-

cient quantization-based multiple-assignment retrieval method discussed in Section 2.3.3.

In Figure 2.10 we consider the trade-off between the retrieval mAP and search speed. The speed is measured as number of distance computations relative to the number needed for exhaustive search. Each curve shows, for a quantization into p cells, the performance using multiple assignment to $m = 1, 2, 4, \ldots, p$ clusters, where $m = 1$ corresponds to the lower-left point of each curve.

The blue curves show performance using 16 clusters: either using a k-means clustering computed over the z_i (solid), or using the GMM clustering used for the local metrics (dashed). Using the GMM clustering (dashed blue curve), a speedup factor 10 relative to exhaustive search can be achieved by single assignment ($m = 1$), but at the cost of a drop of around 10 points in mAP. Using k-means clustering over the $\mathbf{z}_i$ (solid blue curve) we obtain larger speedups and higher mAP values. The results show that it is more effective to dissociate the clustering used for the local metrics from the one used for retrieval, in contrast to the approach taken by Bhattarai et al. [2014].

Moreover, dissociating the clusterings, allows more flexibility in choosing the speed-*vs*.-accuracy operating point. By using k-means clustering with more than 16 centers we can substantially improve the search results: as seen by the red, black, and green curves for p equal to 64, 256, and 1024 respectively. For example, with $p = 1024$ clusters (green curve) and assignment to $m = 64$ clusters we can reduce the search time by a factor 14, without compromising the mAP. Our speedup is comparable to the factor of 10 reported by Bhattarai et al. [2014] for 16 clusters in their hierarchical approach, but our approach leads to better retrieval results. With $p = 1024$ and $m = 8$, a speedup factor larger than 100 can be obtained while loosing less than 5 mAP points.

2.4.3 Experiments for verification

Here, we present the results obtained with CLML for the task of face verification on LFW dataset and compare them to the relevant state-of-the-art. In Table 2.3 we compare our results obtained using local CLML metrics and global LDML ones to the state-of-the-art using the LFW face verification evaluation.

When using no outside training data, the results of Chen et al. [2013] (93.2 ± 1.1) and Simonyan et al. [2013a] (93.0 ± 1.1) are sate-of-the-art. Using the ℓ_2 metric as a baseline we obtain 78.9 ± 0.9, which is improved using global

Using LFW training data		
Guillaumin et al. [2009]	12K	87.5 ± 0.4
Chen et al. [2013]	12K	93.2 ± 1.1
Simonyan et al. [2013a]	12K	93.0 ± 1.1
Ours, FV, without metric learning	12K	78.9 ± 0.9
Ours, FV, LDML, $d=768$	12K	92.3 ± 0.5
Ours, FV, LDML, $d=128$	12K	92.1 ± 0.5
Ours, FV, CLML, $d=128$ $(k=6)$	12K	92.8 ± 0.4
Ours, FV, LDML, $d=32$	12K	91.6 ± 0.4
Ours, FV, CLML, $d=32$ $(k=4)$	12K	92.4 ± 0.5
Using external training data		
Taigman et al. [2014]	4.4M	97.4 ± 0.3
w/o metric learning, 2D alignment	4.4M	94.3 ± 0.4
Yi et al. [2014]	500K	97.7 ± 0.3
w/o metric learning	500K	96.3 ± 0.3
Parkhi et al. [2015]	2.6M	99.0
w/o metric learning	2.6M	97.3
Schroff et al. [2015]	200M	99.6
Ours, CNN, w/o metric learning	500K	96.2 ± 0.8
Ours, CNN, LDML, $d=128$	500K	96.5 ± 1.0
Ours, CNN, CLML, $d=128$ $(k=2)$	500K	96.4 ± 0.9

Table 2.3 – Comparing CLML using FV features with other metric learning methods. Performance as LFW verification accuracy.

LDML metrics to 92.1 ± 0.5 and 91.6 ± 0.4 for $d = 128$ and $d = 32$ dimensional projections respectively. For both projection dimensions, CLML improves over LDML, to 92.8 ± 0.4 and 92.4 ± 0.5 respectively. We also observe a consistent improvement when comparing LDML ($d = 128$ and $d = 768$) with CLML ($d = 32, k = 4$ and $d = 128, k = 6$) using the same number of parameters. This underlines once more that the improvements by CLML are not simply due a larger number of parameters.

Our results differ slightly from those of Simonyan et al. [2013a] due to a more efficient implementation: (i) They used GMMs with 512 components for the FV, while we use only 128, yielding $4\times$ smaller descriptors. (ii) They average over left-right flipped versions of face image which we do not. (iii) Besides a

Mahalanobis metric, they also learn a similarity of the form $\mathbf{x}_i^T \mathbf{M} \mathbf{x}_j$, which they average with the Mahalanobis metric, similar to Cao et al. [2013].

We report results obtained using CNN features in the bottom part of Table 2.3. Using the CNN features with the ℓ_2 metric we obtain 96.2 ± 0.8 verification accuracy, similar to the results of Yi et al. [2014] (96.3 ± 0.3) which used the same training data for their network. Surprisingly, using our CNN descriptors we found that metric learning, either with LDML or CLML, gives only small improvements over the ℓ_2 baseline. The reason for this might be that the performance of the ℓ_2 distance over the CNN features is already very high for the face verification task, or that the pair-wise loss function of LDML is less suitable for verification than the triplet-based loss used by Parkhi et al. [2015], or the weighted chi-squared metric used by Taigman et al. [2014]. Yi et al. [2014] use a multi-task learning objective to train their CNN jointly for both verification and recognition.

The quoted results from the literature other than Yi et al. [2014], are using CNNs trained on datasets that are 5 to 400 times larger, and therefore not directly comparable. Taigman et al. [2014] use 4.4 million images and combine the output of three different CNNs and use 3D face alignment. Using only 2D aligned images (as we do in our work), they reported slightly worse than ours before metric learning (94.3 ± 0.4). Parkhi et al. [2015] recently reported results using a deeper convolutional architecture (Simonyan and Zisserman [2014]) and 2D face alignment over 2.6 million images (99.0). The state of the art results of Schroff et al. [2015] are based on an extremely large proprietary dataset of 200 million images, for which no alignment was used.

2.4.4 Data visualization

To illustrate the benefit of CLML for data visualization we plot LFW images projected using CLML and LDML in Figure 4.21. We learned $d = 256$ dimensional projections on the FV features, and map these to 2D by PCA. For CLML the number of local metrics was set to $k = 12$ by cross-validation. CLML leads to a much better separation of the faces of different people, despite the limited improvement of CLML (81.9) over LDML (80.8) in mAP for $d = 256$. Using CLML we can more clearly see the two groups corresponding to male and female faces. We used the LFW gender labels from the BeFIT website. [2]

2. See http://fipa.cs.kit.edu/431.php

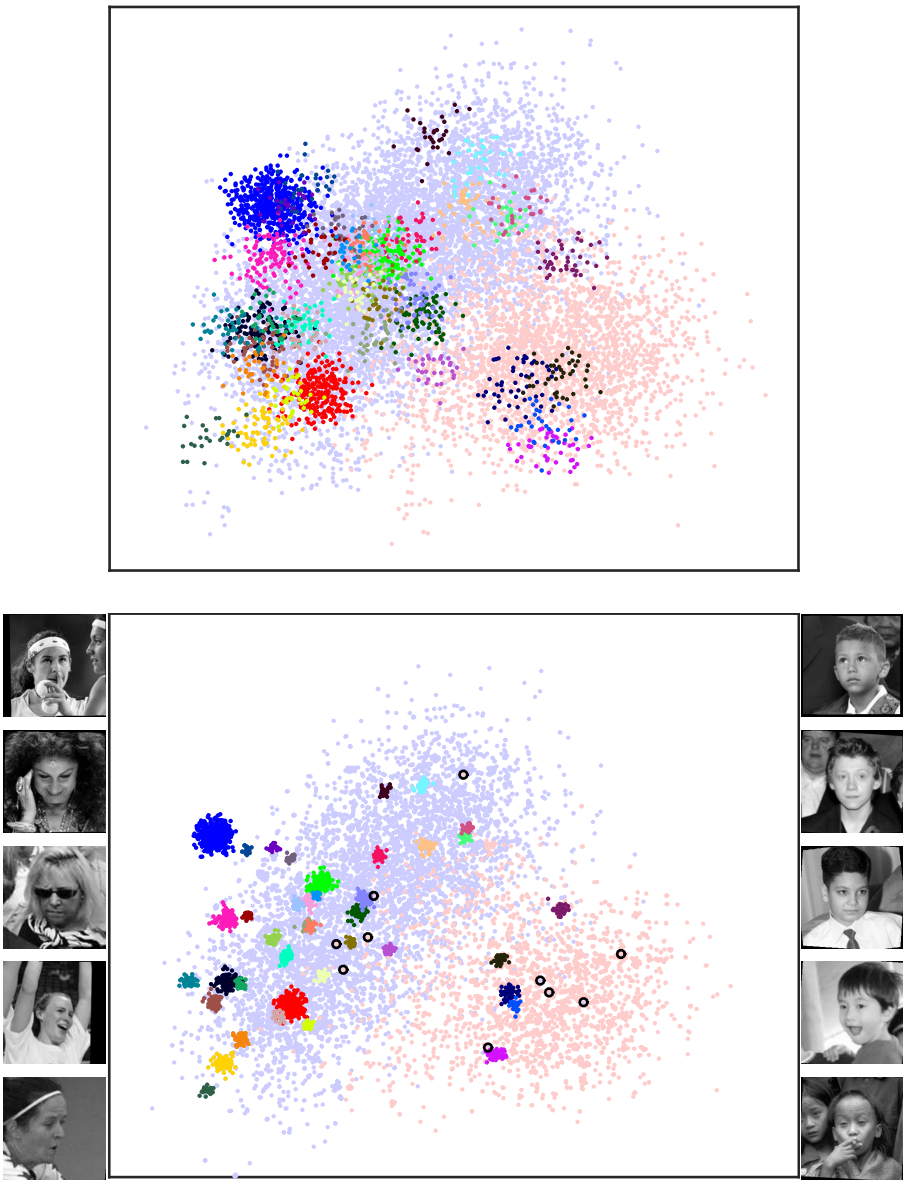


Figure 2.11 – Visualization of the data projections learned by LDML (top) and CLML (bottom). Data points of the 40 most frequent people in the dataset have been color coded. Other data points are plotted in blue and pink for males and females respectively. On the sides of the CLML visualization we show outliers faces (marked with black circles) of males in the female cluster (right), and vice-versa (left). Interestingly, male outlier faces are mostly young boys, while female outlier faces mostly display extreme poses or expressions.

2.5 Conclusion

In this chapter we presented our coordinated local metric learning (CLML) approach which learns local Mahalanobis metrics, and integrates them in a global representation where the ℓ_2 distance is used. This allows data visualization in a single view, and the use of efficient ℓ_2-based retrieval methods. Our low-dimensional global representation is obtained as a linear projection of an expanded data representation, defined using the input data and a Gaussian mixture clustering. One of the attractive properties of our methods is that the CLML metrics can be learned with any global Mahalanobis metric learning method over the precomputed expanded data representation. In our experiments we used LDML as learning method, but other methods that optimize different loss functions can be used.

We have also shown that our proposed method admits different interpretations. CLML can be interpreted as learning a linear projection on top of Fisher vector encoding (corresponding to the mean of the Gaussians) of the input space or as kernel metric learning, where the kernel is a data-adaptive kernel, given by the unsupervised data clustering.

We have validated our approach through extensive experiments for the task of face retrieval on the Labeled Faces in the Wild dataset. We have shown large improvements over earlier work using local metric learning, and that our approach consistently improves over global metric learning using different features, projection dimensions, and performance measures. Our approach also allows efficient multiple-assignment retrieval, which gives a better speed-accuracy trade-off than earlier work for face retrieval in a large-scale dataset with a million distractor faces.

We have also validated our approach for the task of face verification on the Labeled Faces in the Wild dataset. In all settings CLML improves over global LDML metrics, or gives comparable results. We show that the increase in performance of CLML is not due to the increase in parameters, but due to learning a set of coordinated of local metrics. Finally, we have also illustrated the benefit of CLML for performing data visualization.

Chapter 3

Heterogeneous Face Recognition

Contents

3.1 Introduction

Supervised machine learning methods, such as SVM, deep learning, and metric learning have been shown to work very well to learn on a train set, and then generalize to new data. These methods rely on the assumption that the data for training and testing comes from the same underlying distribution. However in practice this assumption is not always true, for *e.g.*, recognizing human face images in near-infrared images while the classifier was trained on visible spectrum

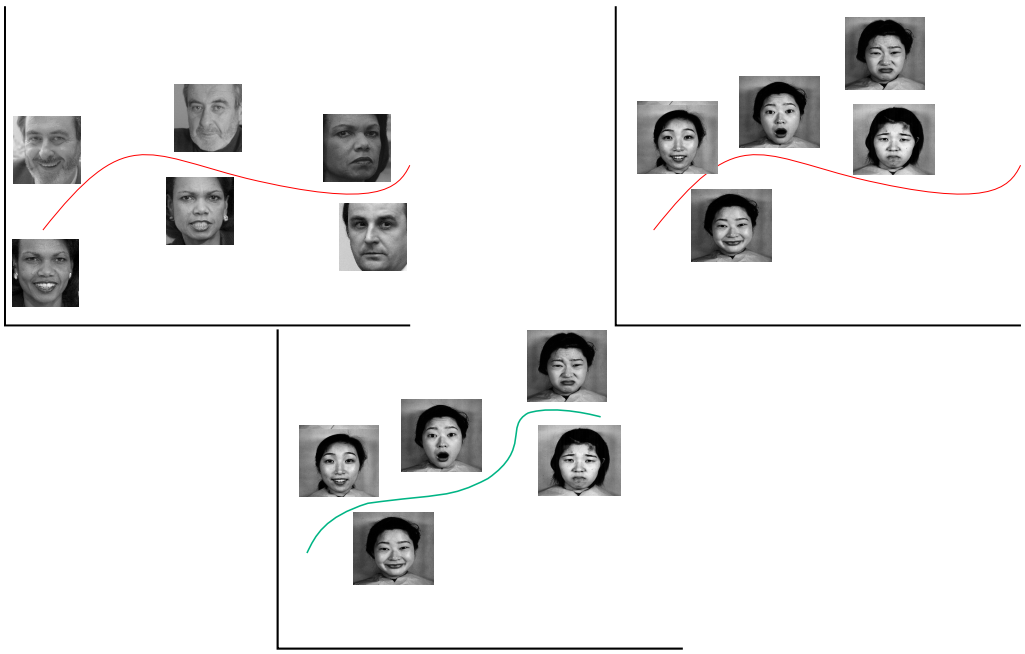


Figure 3.1 – Schematic illustration for domain adaptation. The task is to regress the emotion content of a face image. On left, the red line depicts the regressor learned on the training dataset [Huang et al., 2007]. On right, we show the mismatch between the learned regressor applied directly on images from another dataset [Lyons et al., 1998] comprised of Japanese female faces. In center, we show how domain adapation can used to adapt the regressor to the new target dataset.

image, segmenting organs in MRI images when the algorithm was trained on X-ray images, recognizing day to day objects from images captured with mobile while the classifier was trained on images from online retail websites, *etc*. See Figure 3.1 for an illustration.

In [Torralba and Efros, 2011, Zhou et al., 2014], image classifiers were shown to perform significantly worse when tested on images from other datasets, due to the change in underlying distribution of the data. The difference in the distribution of the datasets can arise from many factors: viewpoint of camera, image resolution, image acquisition device, illumination, *etc*. To address this problem, over the past few years there has been a significant amount of research to answer the question : *"How can we build machine learning pipelines, which can generalize or adapt to a new domain?"*.

This problem is commonly referred as domain adaptation and has been studied in many different areas: statistics and machine learning [Shimodaira, 2000, Daume III and Marcu, 2006, Blitzer et al., 2008, 2011], speech processing [Leggetter and Woodland, 1995, Daumé III, 2007, Blitzer et al., 2006], natural language processing [Jiang and Zhai, 2007, Li et al., 2012] and computer vision [Oquab et al., 2014, Fernando et al., 2013]. Some special kinds of domain adaptation problems have also been studied but under different names including covariate shift [Shimodaira, 2000, Yamada et al., 2012], sample selection bias [Heckman, 1979, Zadrozny, 2004] and class imbalance [Japkowicz and Stephen, 2002]. In this work, we focus on the problem of domain adaptation for the task of heterogeneous face recognition.

Heterogeneous face recognition is the problem of recognizing faces across different modalities. In most cases, the gallery of known individuals consists of normal visible spectrum images. Probe images may be forensic or composite sketches, which are useful in the absence of photos in a forensic context [Klare et al., 2011, Mignon and Jurie, 2012a]. In comparison to the visible spectrum (VIS, $0.38 - 0.70\mu\text{m}$) images, near-infrared (NIR, $0.75 - 1.4\mu\text{m}$) and short-wave infrared (SWIR, $1.4 - 3\mu\text{m}$) images are less sensitive to illumination variation. Mid-wave infrared (MWIR, $3 - 8\mu\text{m}$) and long-wave infrared (LWIR, $8 - 15\mu\text{m}$), also referred to as "thermal infrared", is suitable for non-intrusive and covert low-light and night-time acquisition of face images for surveillance by relying thermal infrared radiation rather than reflection [Kong et al., 2005]. Differences between the gallery and probe modality, make heterogeneous face recognition more challenging than traditional homogeneous face recognition in a single modality.

Visible spectrum face recognition has been extensively studied, and recently much progress has been made using deep convolutional neural networks (CNN) [Parkhi et al., 2015, Schroff et al., 2015, Taigman et al., 2014, Yi et al., 2014]. In part, this progress is due to much larger training datasets. For example, [Schroff et al., 2015] report an error of only 0.37% on Labeled Faces in the Wild (LFW) dataset [Huang et al., 2007], using a CNN trained on a proprietary dataset of 200 million labeled face images. Earlier state-of-the-art work [Simonyan et al., 2013a] used only 10 thousand train images, yielding an error in the order of 7%. Large visible spectrum datasets can be constructed from internet resources, such as IMDb [Yi et al., 2014], or social media websites. This is, however, not possible for IR images or sketches. For the same reason, it is even harder to establish large cross-modal datasets where we have individuals with images in both modalities. In this work, we are interested in the question: *How can*

we leverage the success of CNN models for visible spectrum face recognition to improve heterogeneous face recognition?

It is not straightforward to apply CNN networks trained on visible spectrum images for heterogeneous face recognition, since the image characteristics may differ significantly across the modalities. Also, relatively little training data is available for other modalities than normal visible spectrum images. In our work we propose a solution by using a CNN pre-trained on large visible spectrum dataset (source domain) as a feature extractor. We pre-process the images from different target domains to match the first and second order statistic of the source domain. The pre-processed images are used to extract features from different layers of the pre-trained CNN to give us the image representation. To reduce the modality gap further, we learn projections for source and target domain in order to map them in a common subspace. Once the source and target domain are projected to a common subspace, a common distance measure is available to us, which is used to perform retrieval and verification.

The contribution of this work is to evaluate a number of strategies to use deep CNNs learned from large visible spectrum datasets in conjunction with metric learning to solve heterogeneous face recognition tasks. We evaluate the impact of different design choices including: using feature representations from different layers of CNN, fine-tuning the CNN on target domain, and various forms of metric learning.

To evaluate our approach we use the following setup, which is commonly used in heterogeneous face recognition benchmarks [Chen et al., 2005, Li et al., 2013, Mittal et al., 2014]. We have a limited training dataset which contains images in both domains for a number of individuals, and is used to learn a model to bridge the gap between the domains. The test dataset consists of gallery images in one domain, and probe images in the other domain, and the individuals in the train and test set are mutually exclusive. We train a deep CNN on CASIA WebFace dataset [Yi et al., 2014], which is a large dataset comprised of visible spectrum images. Once the CNN is trained on visible spectrum images, we use it as a feature extractor for representing the face images in source and target domain. We modify the metric learning framework of logistic discriminant metric learning (LDML, [Guillaumin et al., 2009]) to learn a common subspace in order to perform retrieval and verification. Using our proposed approach we obtain results that are on par or better than the state of the art for both VIS-NIR and VIS-sketch heterogeneous face recognition.

Figure 3.2 – Illustration for domain adaptation: Source domain contains the images of the products from Amazon whereas the target domain consists of real life pictures from a camera in an office. The images are taken from Office dataset [Saenko et al., 2010].

Outline The rest of the chapter is organized as follows. In Section 3.2 we discuss a selection of related work which is most relevant to the topics in this chapter. In Section 3.3 we describe the heterogeneous face recognition methods we evaluate in our experiments. The datasets and experimental results are presented in Section 3.4. Finally we present our conclusion in Section 3.5.

3.2 Related work

Supervised machine learning for real world applications faces two main problems: dataset bias and shortage of labelled data for new tasks. Both of them are relevant to different problems, but none the less are related to each other for the task of heterogeneous face recognition as we discuss below.

Since dataset bias exists, we can not learn a classifier on a standard dataset and assume that it would generalize to new tasks and domains. For example, imagine the task of classifying day to day objects, by taking their pictures from mobile phone. To train a classifier for this task, we can obtain rich annotated data from an online retail website *e.g.* Amazon, eBay, etc. Here, even though the object categories to be classified remain same, as it can be seen in Figure 3.2, the data distribution of images from an online website differs from the data distribution of images captured by a mobile phone.

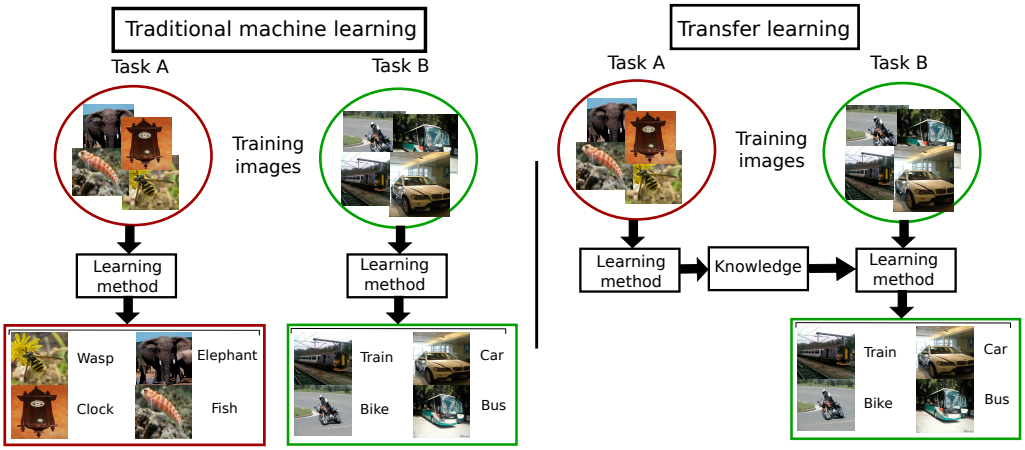


Figure 3.3 – Schematic illustration of difference between traditional machine learning and transfer learning. In comparison to the former, transfer learning re-uses the knowledge across different learning tasks.

Re-training the model from scratch for each new target domain is expensive and requires collecting large set of annotated data. Therefore, in recent years the problem of **domain adaptation** has gained significant attention [Fernando et al., 2013, Gong et al., 2012, Saenko et al., 2010, Gopalan et al., 2011].

Another way to avoid collecting large set of annotated data and training from scratch is to transfer the knowledge between related source and target domains. This line of work is known as **transfer learning** and is inspired from the following observation. Human learners, when faced with new tasks, are able to apply relevant knowledge acquired from solving previous tasks. The more related the new task is to previous tasks, the easier it is for us to solve the new task. Figure 3.3 illustrates the difference between traditional machine learning and transfer learning. Traditional machine learning techniques learn from scratch for each new task. In comparison, transfer learning techniques use the knowledge acquired from solving previous tasks to solve new tasks. For example, the knowledge obtained from training a CNN to classify images can be transferred to other tasks such as object detection, scene classification, visual instance retrieval, *etc*. [Sharif Razavian et al., 2014, Oquab et al., 2014].

Transfer learning and domain adaptation methods make some assumptions about the data distribution mismatch. For domain adaptation, the underlying assumption is that source and target domains are different in terms of marginal data distributions but have identical label sets, see Figure 3.2. In contrast, in

transfer learning, the marginal data distribution as well as the label sets for source and target domain are assumed to be different, see Figure 3.3.

Note that heterogeneous face recognition is both a domain adaptation problem [Patel et al., 2015] (to deal with the discrepancy between the gallery and probe domains), and a transfer learning problem [Pan and Yang, 2010] (to deal with the fact that the subjects/classes in the train and test sets are disjoint).

3.2.1 Overview: Domain adaptation and Transfer Learning

In this subsection we first review some basic terminology used in the context of domain adaptation and transfer learning methods, followed by the relevant related works on each.

Domain adaptation In the context of domain adaptation, there are two domains: **source** domain and **target** domain. We refer to training domain where labelled data data is abundant as the source domain, and the test domain where labelled data is not available, or only very little of it, as the target domain.

Most of the existing domain adaptation based techniques [Fernando et al., 2013, Shekhar et al., 2013, Gopalan et al., 2011, Gong et al., 2012] assume that the source and target domain data use the same feature representation. These techniques fall under the category of **homogeneous** domain adaptation, *i.e*. source and target domain are represented by the same feature space but differ in distribution. In contrast, in **heterogeneous** domain adaptation [Xiao and Guo, 2015, Duan et al., 2012, Wang and Mahadevan, 2011] the difference arises from both the feature space as well as the data distribution. For example, training images for face recognition comes from visible spectrum images, but the testing images are encoded as thermal images of the face. Domain adaptation in this setting is much harder compared to homogeneous domain adaptation.

Transfer learning Transfer learning is one of the ways to address a fundamental issue: shortage of labeled data. As discussed before, often it is the case that the distribution of data changes from one task to another. Re-training the statistical machine learning model from scratch for dealing with distribution change is an expensive and impractical solution given the effort involved in collecting labelled training data. Transfer learning aims to ease the burden of collecting labelled data for the target domain by transferring the knowledge learned in a previous task to a new task [Aytar and Zisserman, 2011, Tommasi et al., 2010].

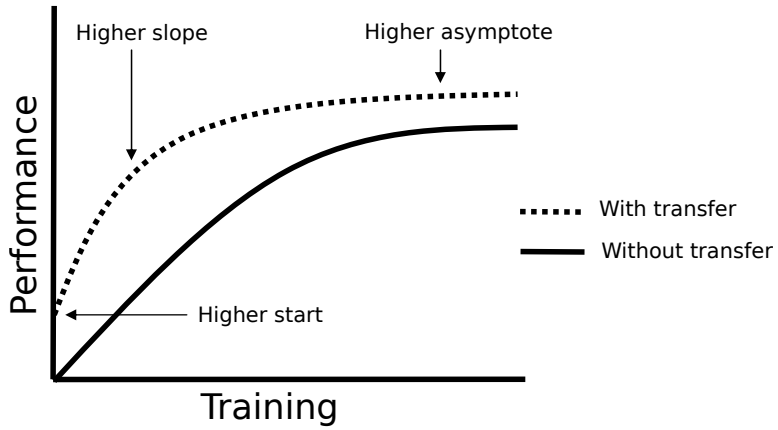


Figure 3.4 – Three ways in which transfer learning might improve learning. Image adapted from Torrey and Shavlik [2009].

The three different ways in which transfer learning might improve the learning are (see Figure 3.4, Torrey and Shavlik [2009]):

- Higher start: initial performance achieved without seeing target domain
- Higher slope: the model is able to learn faster with the help of transfer
- Higher asymptote: the model converges to a higher performance with transfer

Transfer learning has been studied under various names: learning to learn [Thrun and Pratt, 2012], life-long learning [Pentina and Lampert, 2014], knowledge transfer [Mihalkova et al., 2007], etc. Depending upon the specific application, the transferred knowledge can be in the form of instances, feature representation [Ganin and Lempitsky, 2015, Tzeng et al., 2015] or model parameters [Oquab et al., 2014, Babenko et al., 2014, Sharif Razavian et al., 2014]. We refer the reader to [Pan and Yang, 2010] for a more detailed survey of these transfer learning based methods. In this work, we focus on the setting where we transfer the knowledge via model parameters.

As highlighted above, there are many principled solutions to perform domain adaptation and transfer learning. Below we give a detailed overview of subspace based approaches for domain adaptation and deep learning based approaches for transfer learning, since they are the most relevant to the material we present in this chapter.

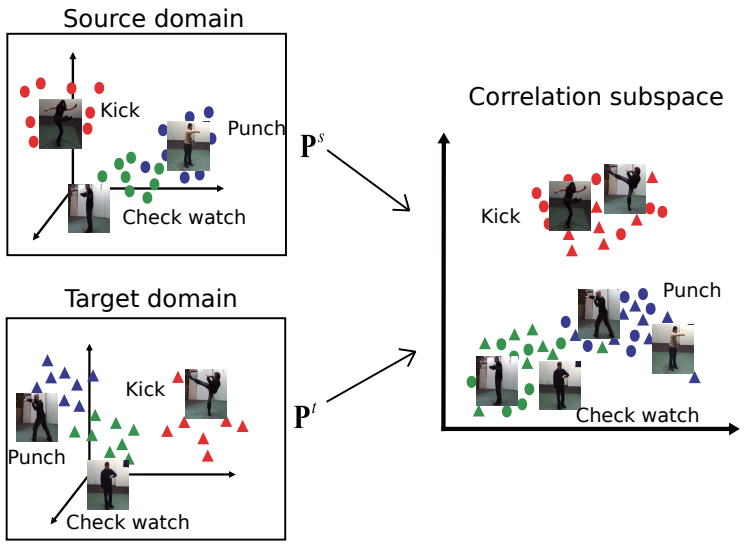


Figure 3.5 – Example of domain adaptation via CCA for the task of action recognition. The source and target domain differ due to the view angle of the camera. $\mathbf{P}^s$ and $\mathbf{P}^t$ denote the projection matrices for the source and target domain obtained via CCA, which maximizes the correlation between the two data distributions. Image adapted from Yeh et al. [2014].

3.2.2 Subspace based approaches

Subspace based approaches aim to derive new feature representations in order to minimize the discrepancy between the source and target domain. These approaches involve learning a subspace, where the discrepancy between the data distributions of the two domains is reduced. In this subspace, both domains share the characteristics, and a classifier learned on the source domain would work on the target domain as well. There are two principled approaches for subspaces based approaches.

In the first set of approaches, one can learn a mapping of one domain to another as a pre-processing step. Doing so makes the two domains comparable and standard machine learning can be applied on top, for example Wang and Tang [2009], Juefei-Xu et al. [2015] maps sketches and near infrared images, respectively, to visible images for the task of face recognition. The second set of approaches involve mapping both domains to a common latent subspace, where the bias due to domain shift is suppressed and the information pertaining to the objects are preserved [Hotelling, 1936, Mignon and Jurie, 2012a, Fernando et al., 2013]. Below we give a brief overview of these methods while limiting our discussion to the state-of-art methods relevant to our work in this chapter.

Among the existing methods, one of the earliest and most effective methods to learn a common subspace is canonical correlation analysis (CCA). Proposed by Hotelling [1936], CCA aims at finding a subspace for projecting two sets of distributions, such that the correlation between the two sets is maximized. CCA seeks a linear transformation for each domain, such that after the transformation the coordinates of both domains (in the common subspace) are maximally correlated. The solution to CCA is obtained via generalized eigen-value decomposition. To deal with non-linearity in the data, kernel methods have also been used for proposing a kernel variant of CCA [Lai and Fyfe, 2000, Melzer et al., 2001, Van Gestel et al., 2001, Hardoon et al., 2004]. Figure 3.5 shows an illustration of CCA for the task of cross-view action recognition.

Fernando et al. [2013] propose an unsupervised domain adaptation method based on subspace alignment. Following the theoretical recommendations of Ben-David et al. [2007], they aim to directly reduce the discrepancy between the two domains by aligning the two subspaces. The source and target domain subspaces $\mathbf{U}$ and $\tilde{\mathbf{U}}$ are composed of the leading eigen-vectors induced by PCA. The objective of their method is to find a transformation matrix $\mathbf{M}$, which aligns the source and the target subspace. They seek the transformation matrix $\mathbf{M}$, such that it minimizes the objective $||\mathbf{UM} - \tilde{\mathbf{U}}||_F^2$, where $||.||_F^2$ represents the Frobenius norm. Fernando et al. [2014] propose an extension of the method so as to obtain a more discriminative source subspace representation. More specifically, they use the label information available in the source domain to learn a metric with pairwise constraints. Once the metric is learned, they project the source data with the linear transformation induced by the learned metric, and perform PCA to obtain a more discriminative source subspace.

Covariance based methods for finding a latent representation like CCA, by construction ignore any negative constraints, *i.e*. constraints given by pairs of non-matching classes. Some works [Yi et al., 2007, Lei and Li, 2009] aim to mitigate this issue by performing linear discriminant analysis (LDA) in each domain, followed by CCA. However, using LDA the maximum dimensionality of the projection is equal to number of classes. Moreover, using the discriminatory information directly for construction of the latent representation might be more effective for learning a discriminative latent subspace. Below we discuss several methods based on this idea, which directly incorporate the label information to find the latent representation.

The work of Saenko et al. [2010] is one of the early works where metric learning was used for the task of domain adaptation. They use ITML [Davis et al.,

2007] to learn a similarity function between $\mathcal{A}$ (source domain) and $\mathcal{B}$ (target domain), optimized to satisfy the pairwise constraints between transformed points. The linear transformation denotes the learned similarity between the two domains:

$$sim_{\mathbf{W}}(\mathbf{x}, \mathbf{y}) = \mathbf{x}^{\top}\mathbf{W}\mathbf{y}, \tag{3.1}$$

where $\mathbf{x} \in \mathcal{A}$ and $\mathbf{y} \in \mathcal{B}$. Note that, such a similarity function can be learned even when the dimensionality of the two domains is different. In comparison to standard metric learning approaches, here $\mathbf{W}$ could be a rectangular matrix and is not constrained to be a positive semi-definite matrix. In their work, they construct the training pairs across the domains and argue that using intra-domain pairs does not model the task of domain adaptation. They show empirically that using both inter and intra domain pairs performs considerably worse as compared to using inter-domain pairs.

Similarly, Mignon and Jurie [2012a] propose cross modal metric learning (CMML) method to extend traditional metric learning approaches to cross-modal problems. They learn a different projection from each modality to a common subspace. They adapt the metric learning objective function of PCCA [Mignon and Jurie, 2012b] to take into account only the cross-domain pairs, which tries to ensure that across the two domains, face pairs of the same person are close, and pairs of different people are far. As compared to methods which require labels of training data points, the advantage of their method is that it only requires cross-domain constraints, for *e.g.* in face verification with photos and sketches, training pairs are photo-sketch pairs of same person, as well as photo-sketch pairs of different persons.

We explore similar metric learning approaches, but explicitly investigate the relative importance of using intra and inter domain pairs, and separate projection matrices. Even though intra-domain pairs are not related to the multi-modal nature of the task, in our work, we show that they can be included in the loss to provide a form of regularization.

3.2.3 Transfer learning based on deep adaptation

Krizhevsky et al. [2012] used convolutional neural networks to obtain a performance leap over the standard image classification methods on the ImageNet 2012 Large-Scale Visual Recognition Challenge (ILSVRC12, [Russakovsky et al., 2015]). This leap was largely due to two factors, rise in GPU computing power and availability of a large set of annotated images. This advancement begs a question: *Will we need to collect 1 million training images for each task ?*

Following this success, there have been works which curate a large dataset for a particular task [Zhou et al., 2014, Babenko et al., 2014, Lin et al., 2014b, Parkhi et al., 2015, Schroff et al., 2015]. For example, Zhou et al. [2014] release a scene centric database with 7 million pictures labelled with scene categories. Using the representations from the CNN trained on their dataset, they outperform the representations obtained from CNN trained on ImageNet dataset by a margin of 10% for the task of scene recognition. Their results highlight the benefit of collecting large datasets tailored for a particular task.

While it is true that the representations obtained from CNN trained on ImageNet dataset suffer from dataset bias and hence might be sub-optimal, it is not cost effective to annotate a large set of images for every task. To address this issue, there have been recent works which explore the use of pre-trained CNNs for transfer learning. Below we give a brief overview of some of these.

Sharif Razavian et al. [2014] advocate the use of features from a pre-trained CNN as a replacement for the hand-crafted features. They use a CNN with architecture from Sermanet et al. [2014] and train it on the ImageNet dataset. They do not fine-tune the net on the target domain, and use the first fully connected layer of the architecture as a feature extractor. They train a linear SVM on top of this feature representation and obtain competitive results for a wide array of tasks such as image classification, scene classification, fine grained classification, attribute prediction and visual instance retrieval.

Oquab et al. [2014] show that a CNN pre-trained on ImageNet dataset can be adapted for image classification on different image datasets. They train a CNN on ImageNet dataset with the network architecture from Krizhevsky et al. [2012]. The architecture contains five convolutional layers followed by three fully connected layers. Once the CNN is trained the penultimate layer of the network is used as a feature extractor. To adapt this feature extractor to a new domain, they append two fully connected layers on top of the feature extractor and fine-tune the extended network on the target domain. The parameters of the feature extractor (pre-trained CNN) are kept fixed and the parameters of the appended layers are the only trainable parameters. Using the proposed method, they obtain near state-of-the art performance on PASCAL VOC 2007 and 2012 datasets [Everingham et al., 2007, 2012] for image classification. Not using any pre-training reduced their performance by 8% indicating the efficiency of the proposed transfer learning approach. Their method is supervised in the sense that they need labelled image data in the target domain. This could be problematic

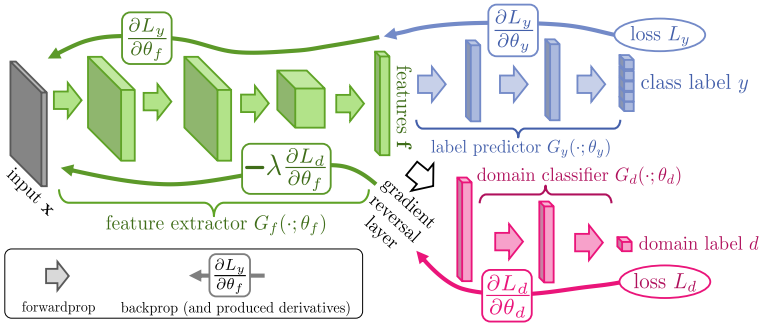


Figure 3.6 – Architecture proposed in Ganin and Lempitsky [2015] for unsupervised domain adaptation. The *feature extractor* (green) and *deep label predictor* (blue) form a standard feed forward neural network. The *Domain classifier* (red) is added on top of feature extractor via gradient reversal layer that multiplies the gradient by a negative constant for backpropagation. Image adapted from Ganin and Lempitsky [2015].

for the cases when there are not many labelled samples available in the target domain.

Similar in spirit to Oquab et al. [2014], Babenko et al. [2014] explore the use of pre-trained CNN architectures [Krizhevsky et al., 2012] for the task of instance retrieval. They train the CNN for classification task and evaluate the features (termed as neural codes) extracted from different layers of the CNN. The main contribution of their work is that they show the performance of a pre-trained net can be improved by re-training the CNN on a dataset similar to the target domain. While they are able to attain state-of-the art performance on a number of datasets, their proposed method might not be feasible in cases where large annotated data similar to target domain is not available.

The shortcoming of the approaches discussed so far is that they need large amount of labelled data in the target domain to fine-tune the CNN. There have been some attempts to train deep architectures with unlabelled or sparsely annotated target data [Ganin and Lempitsky, 2015, Tzeng et al., 2015, 2014]. Below, we provide a brief overview.

Ganin and Lempitsky [2015] propose a novel method for domain adaptation in deep architectures with the use of unlabelled target domain data. Their deep architecture can be broken down into three parts. We illustrate these parts with the help of Figure 3.6. The first part is the *feature extractor* (green) which consists a series of non-linear mappings to output a feature vector $\mathbf{f}$. The second part, *label*

predictor (blue), maps the feature vector $\mathbf{f}$ to the output labels $\mathbf{y}$ via a set of layers. The label predictor and feature extractor form a standard feed-forward net architecture. Finally, the same feature vector $\mathbf{f}$ is mapped to the domain label $\mathbf{d}$ by the third part, *domain classifier* (red). They train the net in order to minimize the label prediction loss on source data and maximize the loss of the domain classifier for both domains. They advocate that the features learned in this way are discriminative for the source domain and at the same time domain-invariant. The advantage of their approach is that they don't need labelled data in target domain. However, the disadvantage is that they need large amount of unlabelled data in the target domain for guiding the training of deep net.

Along similar lines, Tzeng et al. [2015] propose a deep adaptation approach where they minimize the label predictions loss and maximize the loss of the domain classifier. They propose a slight modification in their method, which allows them to train with sparse labelled examples in target domain. Their method is inspired by prior work of "model distillation" [Ba and Caruana, 2014, Hinton et al., 2014]. For each category in the source domain, they compute the average output probability distribution (soft label) over the source training examples. Then, for each target labelled example, they directly optimize to match its predicted distribution over classes with the precomputed soft label, for illustration see Figure 3.7. They argue that doing so, they are able to transfer knowledge about categories which are not explicitly labelled in the target domain. Hence in total, they optimize for three losses, the label prediction loss (over source data), domain classifier loss (over source and target data), and softlabel loss (over labelled target data). One shortcoming of their approach is that it is applicable only when source and target domain share the class labels, a condition which might not always hold.

3.2.4 Heterogenous Face Recognition

In the previous subsections, we gave an overview of different approaches for domain adaptation either via subspace learning or via deep adaptation. In this subsection we give an overview of the work directly relevant to the task of heterogeneous face recognition.

Most work on heterogeneous face recognition falls in one of two families. In the first family, methods are based on reconstructing an image in the gallery domain given an image in the probe domain. In the second family, methods learn a common subspace in which images of both domains are embedded. We discuss these two lines of work in more detail below.

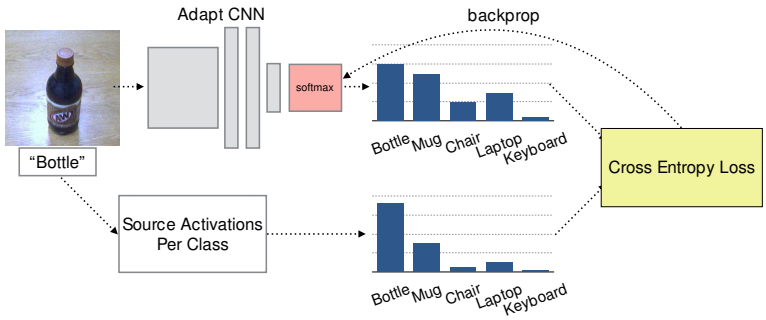


Figure 3.7 – Soft labels in [Tzeng et al., 2015] are computed over the source domain and then used for the cross entropy loss for target domain. Soft labels helps to transfer knowledge about other missing categories, in this case, soft label of bottle induces the knowledge that bottle is more similar to mug than a keyboard. Figure adapted from [Tzeng et al., 2015].

Reconstruction based methods This line of work, see *e.g.* [Sarfraz and Stiefelhagen, 2015, Juefei-Xu et al., 2015], follows a reconstruction based approach that learns a mapping from one modality (typically that of the probe) to the other modality. Once this mapping has been performed, standard homogeneous face recognition approaches can be applied.

Sarfraz and Stiefelhagen [2015] use a representation based on sampling local SIFT features on a dense regular grid in both LWIR thermal and VIS images. A face is represented by concatenating the local descriptors extracted across the sampling grid. They learn a deep fully-connected feedforward neural network to regress the SIFT descriptors in the VIS domain from corresponding descriptors of the LWIR domain. Once the local descriptors in a probe image are mapped to the gallery domain, face descriptors are matched using the cosine similarity.

Wang and Tang [2009] deal with problem of matching sketch to photo images. They learn a multi-scale Markov random field (MRF) model on patches for synthesis of a sketch from a face photo, or vice-versa. Instead of learning the global structure, they aim to learn the local structure by reconstructing local patches. For each patch in a photo, they find similar photo patches in the training set and use their corresponding sketch patches to synthesize sketch of the photo. Their method is based on the assumption that similar face photos have similar sketches. They use patches at multiple scales in conjunction with MRF to incorporate the face structure at different scales and to render smooth sketches. Their method can be used to reconstruct sketches from photos and vice-versa.

Juefei-Xu et al. [2015] propose a method to reconstruct images between NIR and VIS domains, so that the gallery and probe images can be mapped to a particular target domain and matched for face recognition. They learn a dictionary for both domains while forcing the same sparse coefficients for corresponding VIS and NIR images. Doing so allows the use of their coefficients of NIR image to reconstruct the corresponding VIS image and vice-versa.

The advantage of reconstruction-based methods is that it allows the re-use of existing VIS face recognition systems. On the other hand, depending on the discrepancy between the modalities, the problem of cross-modality reconstruction may prove a harder problem than cross-modality face recognition. Photo realistic reconstruction is non-trivial from, *e.g.*, sketches as input, and artifacts in the reconstruction may hurt recognition performance.

Common subspace methods A second line of work learns a mapping from both probe and gallery domain to a common subspace. In this approach matching and retrieval among images from the same or different domains can be performed transparently in the common subspace. The learned subspace is often low-dimensional which is attractive for large datasets to reduce the cost of storage and matching. A more general overview of common subspace methods is provided in Section 3.2.2, here we review work related to heterogeneous face recognition.

Klare and Jain [2010] propose to use Fisher linear discriminant analysis (FLDA) for the task of matching NIR to VIS images. They concatenate LBP [Ojala et al., 2002] and HOG [Dalal and Triggs, 2005] features to give the image representation. Applying FLDA directly on the image representation might be prone to overfitting, and hence they propose to use an ensemble of classifiers trained on random subspaces. They randomly sample (without replacement) a predefined number number of feature vectors to construct the random subspace, and learn FLDA models using both VIS and NIR images. For generating the final representation of the input image they concatenate the projections obtained via FLDA on different random subspaces and perform classification with nearest neighbor classifier.

Crowley et al. [2015] use a triplet-loss similar to LMNN [Weinberger and Saul, 2009] to learn projections to map photos and paintings to a common subspace. The projection aims to ensure that a query photo is closer to a target painting of the same person than to a painting of a different person in the learned common subspace. To represent the faces in photos and paintings, they use Fisher

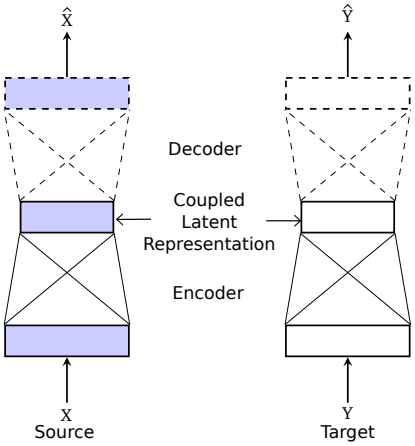


Figure 3.8 – Schematic illustration of coupled autoencoders used for learning a common latent representation. $\mathbf{X}$ and $\mathbf{Y}$ are the source and target domain pair. The solid and dashed lines depict the encoder and decoder network respectively. The parameters of the two autoencoders are not shared.

face representation [Simonyan et al., 2013a] and CNN face descriptors [Parkhi et al., 2015]. They report that a learned distance metric improves the results significantly over the ℓ_2 distance metric when Fisher face representation is used. With the CNN face descriptors, they do not observe further improvements by learning a distance metric. We also use CNN features, but instead of simply using the penultimate network layer, we also investigate the effectiveness of other layers and find these to be more effective.

Riggan et al. [2015] learn a common latent representation with a deep autoencoder. They learn a deep autoencoder for each domain. As depicted in the Figure 3.8, the encoder maps the input to the latent subspace, and decoder uses the latent subspace to reconstruct the input. The parameters for the encoder and decoder of the two autoencoders are not shared. To align the latent representation of the autoencoders, they introduce another term in the objective function, the coupling error. They define the coupling error as the ℓ_2 distance between the latent representation of a source and target domain pair. The parameters for the autoencoders are found by minimizing the reconstruction error as well the coupling error, which ensures that the learned latent representation of two domains is similar.

Jin et al. [2015] learn a set of image filters for reducing the modality gap. The filters are learned in a discriminative manner by utilizing patch level pairwise cross-modality constraints [Mignon and Jurie, 2012a]. At test time, the input

Figure 3.9 – Images of one individual in the CASIA Webface dataset.

images from two domains are processed with the learned filters. The filtered images are divided into patches, and local ternary patterns (LTP, [Tan and Triggs, 2007]) are used for local pattern encoding. Finally, to project these features in a common subspace they use kernel canonical correlation analysis [Hardoon et al., 2004]. Using this approach they obtained state-of-the art results on CASIA-NIR-2 [Li et al., 2013] and CUFSF [Wang and Tang, 2009] datasets.

3.3 Cross-modal recognition approach

In this section we present our method for performing heterogeneous face recognition. First, in Section 3.3.1 we describe our baseline CNN model: it's architecture, training data, feature extraction, and fine-tuning. Then, in Section 3.3.2 we describe how we use metric learning to obtain a subspace representation in which the differences between the modalities are minimized.

3.3.1 Learning a deep CNN model

To train a deep CNN, we use the CASIA Webface dataset [Yi et al., 2014] which contains 500K images of 10,575 individuals collected from IMDb. The images display a wide range of variability in pose, expression, and illumination. Example images of this dataset are shown in Figure 3.9.

We use 100×100 pixel input images to train a CNN with an architecture, detailed in Table 3.1, similar to [Yi et al., 2014] and inspired by the VGG-19 architecture [Simonyan and Zisserman, 2015]. There are a couple of differences in our network, in comparison to the network used in Yi et al. [2014]. The first difference is that we use gray-scale images as input to the network to ensure compatibility with NIR and sketch images. The second difference is that the [Yi et al., 2014] optimize the network for both softmax (identification) and contrastive (verification) loss, whereas in our case we only optimize for the former.

The network is trained to recognize the 10K subjects in the dataset, using the log-loss over the final soft-max layer of the network. Once the CNN is trained,

Name	Type	Size /Stride	Dimension
Conv11	Conv.	$3 \times 3/1$	$100 \times 100 \times 32$
Conv12	Conv.	$3 \times 3/1$	$100 \times 100 \times 64$
Pool1	Max-pool.	$2 \times 2/2$	$50 \times 50 \times 64$
Conv21	Conv.	$3 \times 3/1$	$50 \times 50 \times 64$
Conv22	Conv.	$3 \times 3/1$	$50 \times 50 \times 128$
Pool2	Max-pool.	$2 \times 2/2$	$25 \times 25 \times 128$
Conv31	Conv.	$3 \times 3/1$	$25 \times 25 \times 96$
Conv32	Conv.	$3 \times 3/1$	$25 \times 25 \times 192$
Pool3	Max-pool.	$2 \times 2/2$	$13 \times 13 \times 192$
Conv41	Conv.	$3 \times 3/1$	$13 \times 13 \times 128$
Conv42	Conv.	$3 \times 3/1$	$13 \times 13 \times 256$
Pool4	Max-pool.	$2 \times 2/2$	$7 \times 7 \times 256$
Conv51	Conv.	$3 \times 3/1$	$7 \times 7 \times 160$
Conv52	Conv.	$3 \times 3/1$	$7 \times 7 \times 320$
Pool5	Avg-pool.		320
Class	Softmax		10575

Table 3.1 – Architecture of the CNN learned on visible spectrum gray-scale images of the CASIA Webface dataset. Convolutions (C) use 3×3 filters and stride 1, max-pooling (P) act on 2×2 regions and use stride 2.

we use it to extract features for face recognition experiments on heterogeneous datasets. The feature representation from a layer is obtained by concatenating all activations present in the layer into a single vector. For example, feature representation for P4 layer would be a 12,544 dimensional vector which is equal to the number of activations present in the layer ($7 \times 7 \times 256$). Given a face image we pass it through the network, and extract the features at various layers ranging from C42 to the soft-max layer, unless stated otherwise. Representations extracted from other layers are very high-dimensional and do not improve performance.

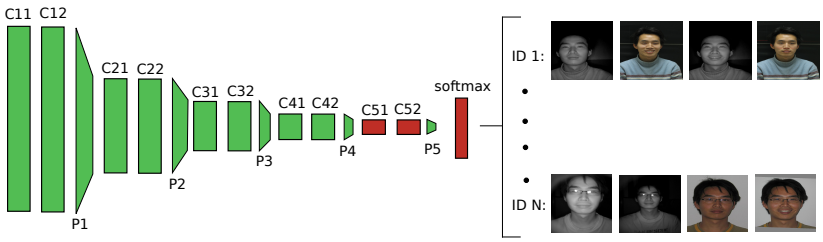


Figure 3.10 – Illustration of CNN fine-tuning. In this schematic example, layers marked in red are fine-tuned to adapt to the heterogeneous dataset. The layers marked in green are frozen, *i.e.* their parameters remain fixed while fine-tuning the network. The network is trained to classify images from both modalities.

We explore fine-tuning the network to adapt to the target domain. We keep the weights fixed throughout the network, except for the topmost soft-max layer, and possibly several preceding layers, see Figure 3.10. The loss function for training is given by the log-loss over a soft-max layer across the subjects present in the given training set. When fine-tuning the model we use images from subjects for which we have images in both modalities. In this manner images of the same subject in the two domains should should be mapped to similar outputs at the final soft-max layer, since images from both domains are used to learn the final classification layer. We also expect features extracted at earlier layers to be comparable, since such features from both domains will be mapped by the same function to the network output.

3.3.2 Metric learning to align modalities

Nuisance factors such as pose, illumination, and expression, make face recognition in uncontrolled settings a challenging problem. The problem is further complicated in heterogeneous face recognition, since images in different modalities differ even if they were acquired at the same moment under the same viewpoint. In single-modality face verification, metric learning has been used extensively used to deal with these difficulties [Guillaumin et al., 2009, Köstinger et al., 2012, Simonyan et al., 2013a, Parkhi et al., 2015, Yi et al., 2014]. Most methods learn a Mahalanobis distance, which is equivalent to the ℓ_2 distance after a linear projection of the data. Supervised training data is used in the form of image pairs of the same person and image pairs of different people. Using this supervision, metric learning techniques can find a projection of the image features that suppresses the effect of nuisance factors, but retains differences due to identity-specific characteristics.

In our work we use logistic discriminant metric learning (LDML) [Guillaumin et al., 2009] to learn Mahalanobis metrics from pairwise supervision. Using $y_{ij} \in \{-1, +1\}$ to denote whether a pair of images $x_i, x_j \in R^D$ are of the same person or not, LDML learns a projection matrix $L \in R^{(d \times D)}$ by minimizing the log-loss

$$\mathcal{L} = \lambda ||L||_F^2 + \sum_{(i,j) \in \mathcal{C}} \ln \left\{1 + \exp\left(-y_{ij}(\tau - d_{ij})\right)\right\}, \tag{3.2}$$

where $d_{ij} = ||L(x_i - x_j)||^2$, and τ is a threshold to classify pairs as being of the same person or not based on the distance of their projections. The set of training pairs is denoted as $\mathcal{C}$. The Frobenius norm regularization term avoids overfitting, and we set λ by cross-validation.

Shared *vs.* separate projection matrices This same approach can also be used in the multi-modal case, by effectively treating the acquisition modality as another nuisance factor. This naive approach requires the use of the same features for both modalities, or at least that they have the same dimensionality. Alternatively, as in [Mignon and Jurie, 2012a], we can treat the examples from the two modalities in a different manner. Let us denote examples from one modality as $x_i \in R^D$ and from the other as $z_j \in R^E$. We can then learn projection matrices A and B to define the cross-modal distance as $d_{ij} = ||Ax_i - Bz_j||^2$, and learn these projections by again minimizing the same log-loss defined in Eq. (3.2). Note that this formulation with a different projection matrix for each domain also allows to learn a common subspace in cases where domain-specific features of different dimensionality are extracted in each domain.

Inter-domain and Intra-domain pairs Another design choice in the metric learning concerns the pairs that are used for training. We make a distinction between intra-domain pairs, which are pairs of images that are both from the same domain, and inter-domain pairs, which consist of one image from each domain. Our goal is to match a probe in one modality with a gallery image of the other modality, the inter-domain pairs directly reflect this in the loss in Eq. (3.2). While the intra-domain pairs are not related to the multi-modal nature of our task, they can be included in the loss to provide a form of regularization and are therefore important to guide the metric learning to learn a projection where the differences between the modalities are reduced.

Figure 3.11 – Images of two individuals in the Labeled Faces in the Wild dataset [Huang et al., 2007].

3.4 Experimental evaluation

In this section we describe the experimental evaluation of the approaches we considered for the task of heterogeneous face recognition. First we present the datasets and evaluation protocols and image pre-processing used in our experiments in Section 3.4.1. Next, we validate the CNN used in our experiments for the task of face verification in Section 3.4.2, followed by evaluation results for heterogeneous face recognition in Section 3.4.3 and Section 3.4.4.

3.4.1 Dataset, protocols, and pre-processing

Labeled Faces in the Wild This dataset [Huang et al., 2007] consists of 13,233 images of 5,749 subjects, see Figure 3.11 for examples. It is probably the most widely used benchmark for uncontrolled face verification. We use it here to validate our baseline CNN model described in Section 3.3.1. Performance is evaluated by the verification performance, where image pairs have to be classified as depicting the same person or not. The classification accuracy is measured across ten different folds, each containing 300 positive and 300 negative pairs. We use the "unrestricted" setting in which all training images may be used for metric learning.

CASIA NIR-VIS This is the largest heterogeneous NIR-VIS face recognition dataset [Li et al., 2013] and contains 17,580 visible spectrum and near-infrared images of 725 subjects. For each individual there are between 1 and 22 VIS images, and between 5 and 50 NIR images. The images present variations in pose, age, resolution, and illumination conditions. In Figure 3.12 we show example VIS and NIR images for one of the subjects in the dataset. We follow the standard evaluation protocol, and set the hyperparameters for our method on

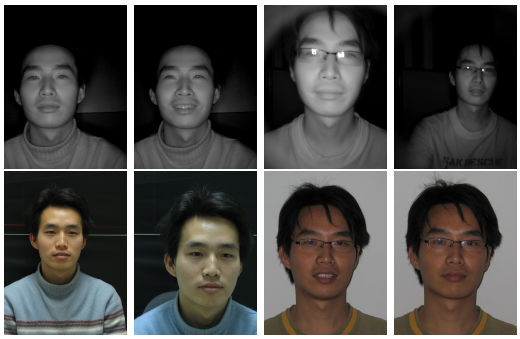


Figure 3.12 – Example NIR (top) and VIS (bottom) images of one individual in the CASIA NIR-VIS dataset.

View1 and report the results on View2, which uses a 10-fold experiment. The individuals in the train and test set are mutually exclusive, and are divided nearly equally among them. The train sets consists of 2,500 VIS images and 6,100 NIR images from 360 subjects. In the test set, the gallery consists of one VIS image for each of 358 subjects, and 6,000 NIR images of the same 358 individuals are used as probes. We report the rank-1 recognition rate, *i.e.* for which fraction of probes the right identity is reported first, and the verification rate (VR) at 0.1% false accept rate (FAR).

ePRIP VIS-Sketch This dataset [Mittal et al., 2014] extends the PRIP dataset [Han et al., 2013] and contains composite sketches for the 123 subjects from AR dataset [Martinez and Benavente, 1998] by adding additional composite sketches. In total there are 4 different types of composite sketches depending upon the software used and the ethnicity of the sketch artist. Due to intellectual property rights only two kind of composite sketches were released. One is made by an Indian artist using the FACES software tool, while the other is made with IdentiKit software tool by an Asian artist[1]. Figure 3.13 shows example face images and corresponding sketches from the dataset. We use the standard evaluation protocol [Mittal et al., 2014]. More specifically, we generate 5 random splits to divide the data into training(48 subjects) and testing (75 subjects) and report the mean identification accuracy at Rank-10.

Face alignment and normalization We align the images in all datasets using a similarity transform, based on facial landmarks that are automatically detected as in [Everingham et al., 2006]. Following [Choi et al., 2012, Sarfraz and Stiefel-

1. See http://www.identikit.net, http:///www.iqbiometrix.com.

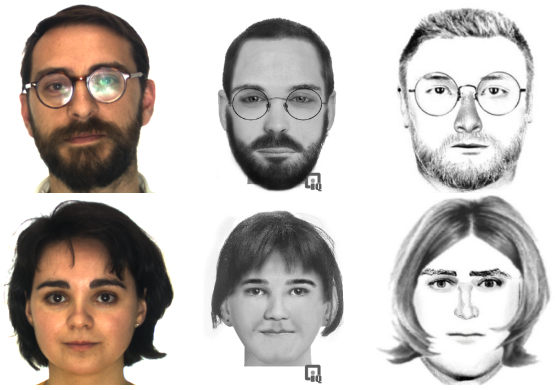


Figure 3.13 – Example images from e-PRIP dataset for two subjects. From left to right: photo, FACES sketch, and IdentiKit sketch.

hagen, 2015], dead pixel values in IR images are replaced by the response of a 3×3 median filter. We also apply an additive and multiplicative normalization, so as to match the per-pixel mean and variance of the CASIA Webface images. For example, for each pixel p in an IR image we set

$$I(p) \leftarrow \frac{\sigma(p)_{WF}}{\sigma(p)_{IR}}\Big(I(p) - \mu_{IR}(p)\Big) + \mu_{WF}(p), \tag{3.3}$$

where $\mu(p)_W F$ and $\sigma(p)_W F$ denote the mean and standard deviation of pixel p across the CASIA Webface dataset, and $\mu(p)_I R$ and $\sigma(p)_I R$ denote the same on the IR dataset. This normalization step is performed for correcting the differences in the first and second order statistics of the signal.

3.4.2 Evaluation on LFW dataset

Before evaluating the performance of CNN features for heterogeneous face recognition, we first validate the effectiveness of our CNN model trained on the CASIA Webface dataset. To this end we evaluate it on the LFW face verfication task.

In this experiment we perform verification on the pairwise distances between face features extracted from several layers of the CNN. We compare the Euclidean ℓ_2 distance, with metrics learned by LDML using the LFW training data, and vary the projection dimension d of LDML (*i.e.* the rank of the metric). Based on the results in Table 3.2 we can make several observations.

1. Metric learning improves performance for all features, but the improvements are small for S and P5.

	$d = 64$	$d = 128$	$d = 256$	Euclidean
S	95.8	95.8	95.8	95.7
P5	95.8	95.9	95.9	95.6
C52	96.7	96.8	96.7	92.8
C51	96.8	96.8	**96.9**	89.9
P4	96.1	96.4	96.3	81.1
C42	95.1	95.5	95.4	75.1

Table 3.2 – Evaluation of LFW verification accuracy using features from different CNN layers, using metric learning to project to different dimensionalities d. We also report results obtained with the Euclidean metric.

2. Which features are optimal depends on whether we use metric learning (C51) or not (S).
3. Metric learning notably reduces the performance differences between the features.
4. Varying the projection dimension from 64 to 256 has only a minor impact on performance. In the remainder of our experiments we therefore use LDML with $d = 64$, unless specified otherwise.
5. The most important observation is that while using only gray scale images, our network (96.9%) performs comparable to that of [Yi et al., 2014] (97.7%) which uses RGB input images.

3.4.3 Results on the CASIA NIR-VIS dataset

We now present an extensive experimental evaluation of heterogeneous VIS-NIR face recognition.

Effect of normalization First, we consider the effect of using the additive and multiplicative normalization. In Table 3.3 we present the results for the two possible configurations. We report these results with metric learning, where for learning the metric we use both inter-domain and intra-domain pairs with a shared projection matrix. From the results we can observe that performing the normalization helps us to boost the performance. In the remainder of our experiments we therefore use this normalization, unless stated otherwise.

Normalization	S	P5	C52	C51	P4	C42
✓	72.6	75.3	80.6	82.9	85.9	84.8
×	66.6	70.4	78.6	80.0	82.4	80.7

Table 3.3 – Evaluation for the impact of normalization on features from different layers of the CNN for the CASIA NIR-VIS dataset. We report results with metric learning.

		S	P5	C52	C51	P4	C42
Inter+Intra	Shared	72.6	75.3	**80.6**	**82.9**	**85.9**	**84.8**
	Separate	66.6	70.4	78.6	80.0	82.4	80.7
Inter	Shared	70.0	74.3	79.8	81.7	83.6	82.0
	Separate	**73.0**	**75.7**	77.9	76.8	76.91	74.7

Table 3.4 – Evaluation of features from different layers of the CNN for the CASIA NIR-VIS dataset for different metric learning configurations. Best results per column highlighted in bold.

Metric learning configurations Next, we consider the effect of (a) using intra-domain pairs in addition to inter-domain pairs for metric learning, and (b) learning a shared projection matrix for both domains, or learning separate projection matrices. In Table 3.4 we present the results for the four possible configurations for various CNN features.

From the results we can observe that using both inter-domain and intra-domain pairs is the most effective to learn a shared projection matrix. For separate projection matrices, it is generally also best to learn from both inter and intra domain pairs. For features of the softmax (S) and fifth maxpool (P5) layer, using separate metrics learned from inter-domain pairs only is best, while for all other features a shared projection matrix learned from inter and intra domain pairs is optimal. This may be explained by the fact that S and P5 features are relatively low-dimensional as compared to the others, and thus less likely to be affected by overfitting when learning separate projection matrices. The overall best results are obtained using a shared projection matrix learned from intra and inter domain pairs on P4 features. Unless stated otherwise, this is the setting we use in the experiments below.

VIS feature \ NIR feature	S	P5	C52	C51	P4	C42
S	72.6	71.0	75.0	74.5	72.6	71.1
P5	69.1	75.3	76.5	75.9	74.7	73.1
C52	70.6	73.8	80.6	78.4	77.7	77.5
C51	70.9	74.4	79.0	82.9	80.3	79.0
P4	69.6	72.8	77.9	80.5	85.9	81.2
C42	68.0	71.0	77.6	79.2	81.1	84.8

Table 3.5 – Combining different features for the VIS (rows) and NIR (columns) domain of the CASIA NIR-VIS dataset. Best results per VIS feature highlighted in bold.

Combining different features The optimal features might be different depending on the modality. Therefore next, we experiment with combining different features for the two domains. In this case we learn separate projection matrices, since the feature dimensionalities may differ across the domains. In Table 3.5, we show results for each combination of choosing among six features in each domain. The results show that by learning different projection matrices, we can effectively mix different features. For example, using S or P5 features in VIS domain, we can best combine these with C52 features in the NIR domain. For the case where we have same features in both domains, *i.e*. the diagonal of the table, the feature dimensionality is the same and hence we use a shared metric.

The best results, however, are obtained by using P4 features in both domains. Therefore, we will use the same feature in both domains in further experiments.

Fine-tuning So far we have evaluated different settings with features extracted from a net pre-trained on CASIA WebFace dataset. Now, we evaluate the effect of fine-tuning the pre-trained CNN using the training data of the CASIA NIR-VIS dataset. In our evaluation, once the CNN is fine-tuned, we extract the feature representations from different layers of the CNN and learn a metric on top. In Table 3.6, we report the results with metric learning, where we use both Inter+Intra pairs with a shared metric. The results show that fine-tuning improves the S, P5, and C52 features. If we fine-tune up to C51 or deeper (not shown), the features deteriorate w.r.t. the pre-trained net, probably because at this point fine-tuning

Features used	Finetuning layers: None	S	S+C52	S+C52+C51
S	72.4	75.3	**76.7**	58.3
P5	75.2	75.2	**79.8**	60.6
C52	81.1	81.1	**82.8**	65.6
C51	**83.0**	**83.0**	**83.0**	67.1
P4	**85.4**	**85.4**	**85.4**	**85.4**
C42	**84.5**	**84.5**	**84.5**	**84.5**

Table 3.6 – Fine-tuning to different depths (columns) on the CASIA NIR-VIS dataset, while extracting features from different layers (rows). Best results per feature highlighted in bold.

overfits on the limited training data that is used. Finetuning a CNN over 10 folds (View 2) for all the different settings was computationaly expensive, therefore here results are reported on the development set (View 1).

We also experimented with fine-tuning the softmax layer of the network separately for each modality, and then combining the features using a shared projection learned from both inter and intra domain pairs. This improved results from 72.4 to 74.0 w.r.t. the pre-trained network. It is, however, worse than the 75.3 we obtained by fine-tuning the *Softmax* layer using data from both domains.

The best results, however, are obtained with the P4 features extracted from the pre-trained net (85.9). In the remainder of the experiments we do not use any fine-tuning.

Comparison to the state of the art In Table 3.7 we compare our results of the (Shared, Inter+Intra) setting to the state-of-the-art unsupervised domain-adaptation approach of Fernando et al. [2013], and a ℓ_2 distance baseline that uses the raw CNN features without any projection. From the results we can observe that our supervised metric learning results compare favorably to the results obtained with unsupervised domain adaptation. Moreover, we find that unsupervised domain adaptation improves only marginally over the raw features. This shows the importance of using supervised metric learning to adapt features of the pre-trained CNN model to the heterogeneous face recognition task.

	S	P5	C52	C51	P4	C42
Raw	63.1	62.7	63.8	51.0	29.4	26.8
Our proj.	72.6	75.3	80.6	82.9	**85.9**	84.8
Domain adapt.	63.1	62.7	64.2	51.8	31.8	28.6

Table 3.7 – Comparison on CASIA NIR-VIS using raw CNN features, our projections, and domain adaptation [Fernando et al., 2013]. For the latter, the projection dimension is set on the validation set.

	Rank-1	VR at 0.1% FR
[Li et al., 2013]	23.7 ± 1.9	-
[Riggan et al., 2015]	33.1 ± 6.6	-
[Dhamecha et al., 2014]	73.3 ± 1.1	-
[Jin et al., 2015]	75.7 ± 2.5	55.9
[Juefei-Xu et al., 2015]	78.5 ± 1.7	**85.8**
[Lu et al., 2015]	81.8 ± 2.3	47.3
[Yi et al., 2015]	**86.2 ± 1.0**	81.3
Ours	85.9 ± 0.9	78.0

Table 3.8 – Comparison of our results with the state of the art on CASIA-NIR dataset.

In Table 3.8 we compare our results to the state of the art in our best setting (P4, Shared, Inter+Intra). For the identification experiments, we obtain (85.9 ± 0.9) rank-1 identification rate which is comparable to the state of the art reported by Yi et al. [2015] (86.2 ± 1.2). Yi et al. [2015] extract Gabor features at some localized facial points and then use a restricted Boltzman machine to learn a shared representation locally for each facial point. They process the descriptors using PCA and report worse results using metric learning. Our approach is quite different from them, since we do not learn our feature representations on the CASIA-NIR dataset rather we only learn a metric on top of features from a pre-trained CNN.

For the verification experiments, the state of the art is reported by Juefei-Xu et al. [2015] (85.8 %), followed by Yi et al. [2015] (81.3 %), and our verification

		S	P5	C52	C51	P4	C42	C41	P3
Faces(In)	Inter + Intra	46.7	44.3	48.3	53.3	61.3	58.1	64.8	**65.6**
	Inter	45.6	42.7	46.7	55.5	65.6	58.9	64.3	63.7
IdentiKit(As)	Inter + Intra	27.7	30.4	27.7	38.7	49.6	48.0	50.9	51.5
	Inter	25.1	26.4	28.8	38.1	47.5	44.0	**52.0**	49.6

Table 3.9 – Rank-10 identification accuracy on the e-PRIP composite sketch database. Best result per dataset highlighted in bold.

rate of 78.0 %.

3.4.4 Results on the ePRIP VIS-Sketch dataset

In this section we report our experiments on the ePRIP sketch datasets. We do not evaluate fine-tuning of the CNN on this dataset due to the small size of the training dataset.

Metric learning configurations In Table 3.9 we report the results for two types of composite sketches: Faces(In) and IdentiKit(As). We consider the effect of using intra-domain pairs in addition to inter-domain pairs for metric learning. As before, we use a shared projection matrix in all settings.

In these experiments there it is not clear cut whether adding intra-domain pairs is beneficial. This might be due to the fact that there are only 96 training images in each domain in this dataset. Averaged over Faces(In) and IdentiKit(As), P3 features and both intra and inter domain pairs gives the best results. This similar to the best setting for the CASIA NIR-VIS dataset, except that there P4 was better. The fact that here deeper CNN features are better may be related to the fact that in this dataset, the domain shift is relatively large compared to CASIA NIR-VIS dataset.

In Table 3.14 we compare our results to the state of the art on the e-PRIP dataset. We obtain the best performance on the Faces(In) sketches, outperforming the previous best of [Mittal et al., 2015] by 5%. For the IdentiKit(As) sketches our results are on par with those reported by [Mittal et al., 2015].

In Figure 3.15 we plot the Cumulative Match Characteristic (CMC) curve for our method compared to the existing approaches on Faces(In) dataset, curves for

	Faces(In)	IdentiKit(As)
[Bhatt et al., 2012]	24.0 ± 3.4	15.4 ± 3.1
[Mittal et al., 2014]	53.3 ± 1.4	45.3 ± 1.5
[Mittal et al., 2015]	60.2 ± 2.9	$\mathbf{52.0 \pm 2.4}$
Ours	$\mathbf{65.6 \pm 3.7}$	$\mathbf{51.5 \pm 4.0}$

Figure 3.14 – Rank-10 identification accuracy on the e-PRIP composite sketch database (left), and CMC curve for the Faces(In) database (right) for our result reported in the table.

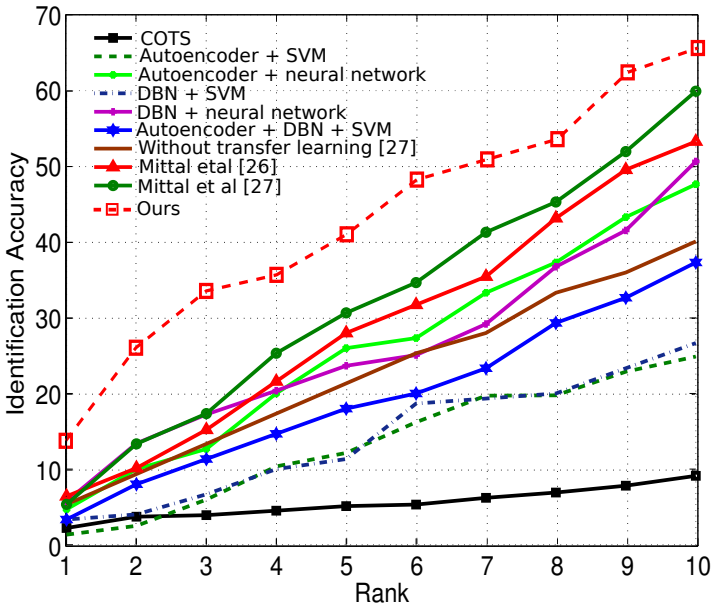


Figure 3.15 – CMC curve for e-PRIP composite sketch database Faces(In) for our result reported in Table 3.14.

other methods are taken from [Mittal et al., 2015] . The figure shows that we obtain significant gain at all ranks compared to the existing state of the art.

3.5 Conclusion

In this chapter we considered how to leverage CNNs pre-trained on large scale visible spectrum images for the task of heterogenous face recognition. We use the pre-trained CNN as a feature extractor along with metric learning for solving the task of heterogenous face recognition. We studied different aspects of our proposed method: extracting features from different CNN layers, finetuning the CNN, and using various forms of metric learning.

We evaluate the impact of different design choices by means of extensive benchmark results on visible to near-infrared and visible to sketch recognition. In our experiments, we found fine-tuning the CNN to the target domain gave limited success. The best results were found by using the layers of pre-trained CNN as feature extractors coupled with metric learning. For our experiments, we pre-process the images of the target domain, so as to match the first and second order statistics of the source domain (visible spectrum images). One of the advantages of our proposed method is that we need to train the CNN only once on the visible spectrum image dataset, and later we are able to adapt it for different domains and datasets. This property is quite attractive, as it allows for model reuse and keeps the computational footprint low. Our solution is quite simple and elegant, and achieves competitive results with the state-of-the art.

For LFW, CASIA NIR and ePRIP dataset, we found the optimal features to be present at C51, P4 and P3 layers of the pre-trained CNN respectively. This suggests that, for a given modality, the depth of optimal features in a CNN pipeline is positively correlated with the domain shift from the CNN training data. Using features from the different layers of pre-trained CNN coupled with metric learning we obtain state of the art results on e-PRIP dataset and competitive with the state of the art on the CASIA-NIR dataset.

Chapter 4

Convolutional Neural Fabrics

Contents

4.1 Introduction

The discriminative power of feature descriptors, and their invariances to nuisance factors, play a key role in computer vision tasks such as matching for 3D reconstruction from images [Snavely et al., 2006], large scale image retrieval

and classification [Philbin et al., 2007, Perronnin et al., 2010], *etc*. During the last decade, a significant amount of work in computer vision has been dedicated to develop discriminative feature descriptors. Out of these, SIFT [Lowe, 1999], HOG [Dalal and Triggs, 2005] and LBP [Ojala et al., 2002] are some of the descriptors which were adopted for a wide use in the community. The majority of these feature descriptors are hand-crafted, though some exceptions [Philbin et al., 2010, Brown et al., 2011] exist, where machine learning is used to learn them in a supervised setting.

Traditionally, a computer vision algorithm consists of two independent pipelines. In the first step, a feature descriptor is used to characterize the content of an image at different spatial locations, and this information is encoded into an image descriptor. In the next step, machine learning techniques are built on top of the image descriptor to increase the invariance to various nuisance factors. In the past years, a lot of progress has been made on both fronts and has lead to impressive results for a variety of computer vision tasks, for *e.g*. image classification [Perronnin et al., 2010], image retrieval [Jégou et al., 2011, Arandjelovic and Zisserman, 2013], face verification [Simonyan et al., 2013a], *etc*.

Despite this progress, the traditional approach has two major shortcomings. First, the optimum feature descriptor for each new task needs to be hand-crafted. Second, in the majority of computer vision algorithms these two pipelines, the machine learning techniques and feature encoding methods, are optimized independently and hence might be suboptimal.

Convolutional neural networks (CNNs) [LeCun et al., 1989] have been used to learn the feature representations alleviating the need for handcrafting feature descriptors and have proven extremely successful for a wide range of computer vision problems and other applications. In particular, the recent results of Krizhevsky et al. [2012] have caused a major paradigm shift in computer vision from models relying in part on hand-crafted features, to end-to-end trainable systems from the pixels upwards. This has been a welcome change, since for any given task the features do not require to be hand-crafted and can be learned as a part of the CNN architecture. The CNN unifies the machine learning and feature descriptor pipelines, and optimizes them jointly.

In the recent years, a lot of progress has been made in designing efficient architectures for image segmentation and image classification. One of the main problems that holds back further progress using CNNs, as well as deconvolutional variants [Noh et al., 2015, Ronneberger et al., 2015] used for semantic

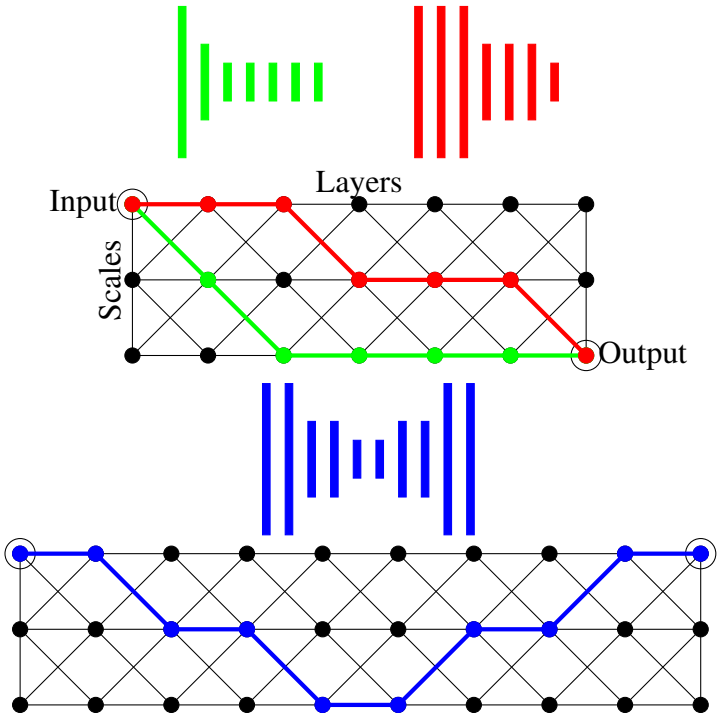


Figure 4.1 – Trellis embedding of two seven-layer CNNs (red, green) and a ten-layer deconvolutional network (blue). Feature map size of the CNN layers are given by height. Layers of the trellis are laid out horizontally, scales vertically. Trellis nodes receiving the input and producing output are encircled. All edges are oriented to the right, down in the first layer, and towards the output in the last layer. The channel dimension of the 3D trellis is omitted for clarity.

segmentation, is the lack of efficient systematic ways to explore the discrete and exponentially large architecture space. To appreciate the number of possible architectures, consider a standard chain-structured CNN architecture for image classification. The architecture is determined by the following hyper-parameters: (i) number of layers, (ii) number of channels per layer, (iii) filter size per layer, (iv) stride per layer, (v) number of pooling *vs*. convolution layers, (vi) type of pooling operator per layer, (vii) size of the pooling regions, (viii) ordering of pooling and convolution layers, (ix) channel connectivity pattern between layers, (x) type of activation, *e.g*. ReLU or MaxOut, per layer. The number of resulting architectures clearly does not allow for (near) exhaustive exploration. Instead of manually designing compact architectures with maximum performance, our goal is to automate the current practice of hand-crafting architectures for certain tasks. Our neural fabrics circumvent eight of the ten CNN architecture-related hyperparameters highlighted above.

In this chapter, we show that all CNN and deconvolutional architectures that can be obtained for various choices of the above ten hyper-parameters are embedded in a "fabric" of convolution and pooling operators. Concretely, the fabric is a three-dimensional trellis of response maps of various resolutions, with only local connections across neighboring layers, scales, and channels. See Figure 4.1 for a schematic illustration of how the trellis embeds different architectures. Each activation in the trellis is computed as a linear function followed by a non-linearity from a multi-dimensional neighborhood (spatial/temporal input dimensions, a scale dimension and a channel dimension) in the previous layer. Setting the only two hyper-parameters, number of layers and channels, is not critical as long as they are large enough. We also consider two variants, one in which the channels are fully connected instead of sparsely, and another in which the number of channels doubles if we move to a coarser scale. The latter allows for one to two orders of magnitude more channels, while increasing memory requirements by only 50%.

All chain-structured (de)convolutional architectures embedded in the fabric can be recovered appropriately setting certain weights to zero, so that only a single activation path is non-zero along the scale and layer axes. General non-path weight settings correspond to ensembling many architectures together, which share parameters where edges overlap. The acyclic trellis structure allows for learning using standard error back-propagation methods. Learning can thus efficiently configure the fabric to implement each one of exponentially many architectures and, more generally, ensembles of all of them. Experimental results competitive with the state of the art validate the effectiveness of our approach.

Our contributions are: (1) Our fabric allows to by and large sidestep the CNN model architecture selection problem. Avoiding explicitly training and evaluating individual architectures using, *e.g.*, greedy local-search strategies [Chen et al., 2016]. (2) While scaling linearly in terms of computation and memory requirements, our approach leverages exponentially many chain-structured architectures in parallel by massively sharing weights between them. (3) Since our fabric is multi-scale by construction, it can naturally generate output at multiple resolutions, *e.g.* for image classification and semantic segmentation, by connecting a prediction layer to nodes at different scales in the last layer. without requiring to design ad-hoc branching architectures for this purpose.

We validate our approach on CIFAR10 [Krizhevsky, 2009] and MNIST [LeCun et al., 1998] dataset for image classification, and PartLabels dataset [Kae et al., 2013b] for image segmentation. In our experiments, we explore the per-

formance of the trellis by varying the number of layers and channels used in the architecture. We learn the parameters of the trellis with standard SGD with momentum and obtain results competitive with the state of the art on all three datasets. We also visualize the weights of the learned trellis to explore the topology of learned trellis and observe qualitative differences between the models learned for the three datasets.

Outline The rest of the chapter is organized as follows. First in Section 4.2 we give a brief overview of CNN followed by discussion of related work most relevant to the topics presented in this chapter. In Section 4.3 we present the construction of our trellis model along with implementation details. We present extensive experimental results in Section 4.4 detailing the effect of the few hyperparameters of our trellis model. We also compare our results to the current state-of-the art for different tasks like image classification and segmentation. Finally, we conclude this chapter in Section 4.5.

4.2 Related work

In the recent years, there has been a considerable amount of research in the field of convolutional neural networks (CNN) for computer vision tasks. The chain-structured CNN architecture used by Krizhevsky et al. [2012] for ImageNet classification is widely used for many other vision tasks [Donahue et al., 2014, Sharif Razavian et al., 2014]. The tasks range from image classification [Krizhevsky et al., 2012] to image retrieval [Babenko et al., 2014] to image segmentation [Long et al., 2015], *etc*. Other widely adopted architectures are the VGG16 and VGG19 networks of Simonyan and Zisserman [2015]. Although effective for many tasks, it is by no means clear that these architectures are among the best ones given their computational and memory requirements. Their widespread adoption is at least in part due to the lack of more effective techniques to find good architectures than extremely costly brute-force exhaustive or local search [Chen et al., 2016].

Below we first review some basic terminology and notation for CNN, followed by a brief overview of different hand-crafted CNN architectures for the problems of image classification and segmentation.

4.2.1 Convolutional neural network

A convolutional neural network (CNN) [LeCun et al., 1998, 1989] consists of a series of convolution and pooling layers, optionally followed by fully con-

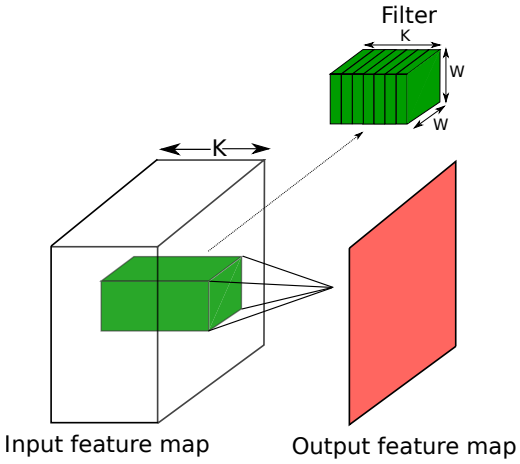


Figure 4.2 – Illustration of the convolution operator inside the convolution layer of a CNN. A filter (dark green) is convolved over different locations of the input feature map. At each location of the input feature map, the dot product of the filter and the entries in the feature map (light green) gives the output response (light red) for the corresponding location in the output feature map.

nected layers. A CNN transforms an input through a series of hidden layers. Each hidden layer consists of a linear transformation (*e.g*. convolution for a convolution layer and dot-product for a fully connected layer) to be applied on its input, optionally followed by a non-linearity (*e.g*. ReLU, sigmoid, *etc*.).

The input and output of a hidden layers are set of arrays called feature maps. For example, for the first layer of a CNN with a grayscale image as an input, the input feature map would be a 2D array (for a video it would be a 3D array, and for an audio or DNA sequence it would be a 1D array). In comparison to a standard neural network, the CNN architecture assumes the input to have spatially-local correlations. These assumptions are encoded in the CNN architecture so as to have an efficient implementation. Below we give a brief description about the standard layers used in a CNN, including, but not limited to, convolution, fully connected, pooling and upsampling followed by a note on activation functions.

Convolution layer A convolution layer consists of a set of learnable filters and biases. Each filter has a small spatial support region but extends through the entire depth of incoming feature maps. For example, in Figure 4.2, the filter has a size of $W \times W \times K$, where $W \times W$ is the spatial support of the filter, and K is the number of input feature maps.

In the forward pass of a CNN, we slide the filter over the different locations of the input feature map and compute the dot product with the entries of the feature map. This operation, generates an activation for each location we convolve, providing us with an output feature map. Intuitively, the filter convolves over the input feature maps, to search for a particular pattern (*e.g*. edges, corners, faces, *etc*.) and this is reflected in the magnitude of the activation present in the output feature map. Repeating this process with a set of C filters, we can compute C output feature maps, each one computed independently. Once the output feature maps are computed, a learnable bias, specific to each output feature map, is added to the corresponding map. There are couple of hyperparameters which one has to set for a convolution layer, below we give a brief detail about each.

The filter size corresponds to the spatial support of the filter. In general practice, the filters are symmetric in shape and 3×3 is the most commonly used filter size. Some recent works explore the use of asymmetric filters [Szegedy et al., 2015b].

The number of filters corresponds to the depth of the output volume, *i.e*. the number of output feature maps produced. The computational and memory footprint of a convolution layer scales linearly with the number of filters used.

The next hyperparameter is the stride with which we slide our filter on the input feature map. A stride of 1 corresponds to moving the filter 1 pixel at a time, whereas a stride of 2, jumps the filter 2 pixels at a time, producing a spatially downsampled feature map .

Zero padding corresponds to the amount of zero pixels padded around an input feature map. Padding allows us to control the size of the output feature map, and is most commonly used to preserve the spatial size of the input feature map. For example, with a filter size of 3×3, a zero padding of 1, would result in an output feature map with the same spatial size as the input feature map.

Fully connected layer A fully connected layer of a neural network connects all the activations in an input feature map to each activation in the output feature map. The output activations in this case can be computed directly with a dense matrix multiplication.

The fully connected layers are similar to the convolution layers, except that the latter learns a function over a small spatial support and the shares the parameters

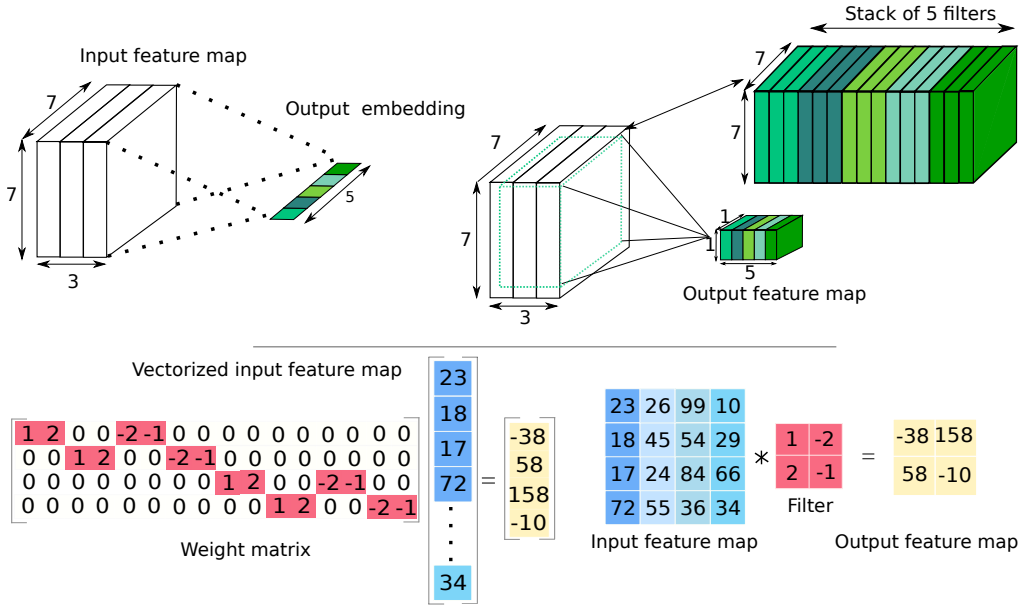


Figure 4.3 – In this figure we depict the equivalence between a convolution and a fully connected layer. In the top, we interpret the 5 dimensional output embedding of a fully connected layer as an output feature map (5 channels and 1×1 spatial size) of a convolution layer. See text for more details. In the bottom, the weight matrix of a fully connected layer is shown, which implements the convolution operation of a 2×2 filter when convolved with stride 2.

over different locations on the input feature map. Apart from this, both layers compute linear functions. Therefore,

1. Any fully connected layer can be re-written as a convolution layer. We illustrate this in Figure 4.3 (top), where the fully connected layer takes a $7 \times 7 \times 3$ feature map as an input and maps it to a 5 dimensional output embedding. The 5 dimensional embedding can be equally interpreted as an output feature map (5 channels with 1×1 spatial size) of a convolution layer. Parameters of the fully connected layer are casted as parameters of a convolution layer, *i.e.* as a stack of 5 filters each with 7×7 filter size . Note, in this case, the filter size is equal to the size of the input feature map.
2. Equivalently, any convolution layer with a particular filter can be implemented as a fully connected layer which implements the same function. The weight matrix of the fully connected layer would be large and sparse (due to small spatial support) and would have identical parameters at different places (due to parameter sharing), see Figure 4.3 (bottom).

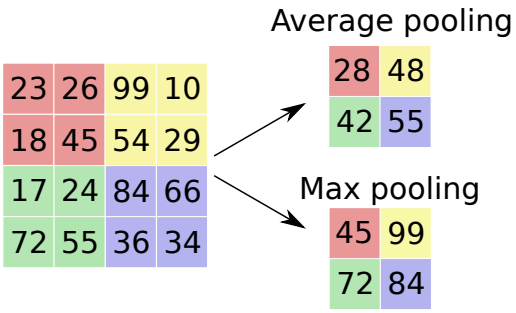


Figure 4.4 – Illustration of max and average pooling with stride and filter size set to 2 and 2×2 respectively.

Out of the two conversions discussed above, the former is particularly useful as it allows to interpret the entire network as convolutional in nature. This makes it possible to evaluate a CNN over images of varying size. Long et al. [2015] used this conversion for performing semantic segmentation on images.

Pooling layer The pooling layers are responsible for downsampling the spatial size of the feature map. They operate independently on each input channel of the feature map, and resizes them spatially. One of the most widely used form of pooling is max-pooling. Max-pooling with a filter size 2×2 applied with a stride 2, would downsample the input feature map by factor 2 in spatial dimensions. A particular value in the output response map is computed by taking a max over 4 values in the input feature map, see Figure 4.4. Average pooling is another variant of pooling which is used in CNN's.

Upsampling layer As the name suggests, upsampling layers are responsible for upsampling the spatial resolution of the input feature maps in a CNN architecture. Long et al. [2015] use bilinear interpolation for upsampling the feature maps. The bilinear interpolation can be interpreted as a convolution of a dilated feature map with a pre-defined filter. We illustrate this operation in Figure 4.5. As it can be seen in the figure, the upsampling layer upsamples the feature map by factor 2. Upsampling by larger powers of 2 can be obtained by cascading multiple upsampling layers. The upsampling layer can either be fixed to perform bilinear interpolation [Long et al., 2015], or initialized to perform bilinear interpolation, and then learned.

Activation function An activation function is an element-wise function applied on the outputs of a hidden layer (convolution or fully connected). In its simplest form, the activation function could be an identity function *i.e.* $f(x) = x$.

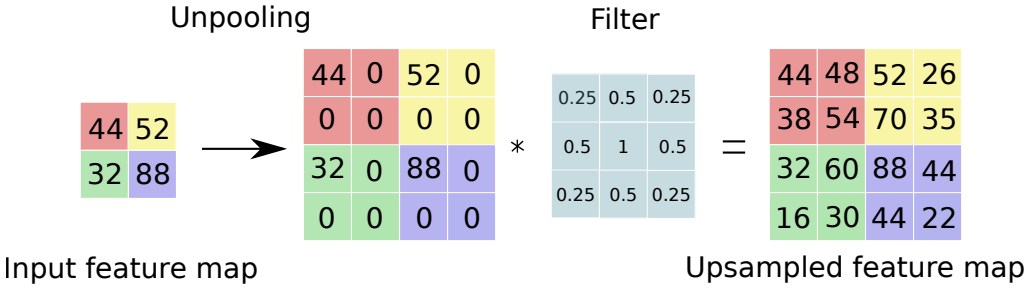


Figure 4.5 – Illustration of upsampling operation for upsampling the spatial size of a feature map. The first step involves doing a simple top-left unpooling operation. Next step involves convolution with a filter whose weights are set to perform bilinear interpolation. In our work, the weights of the filter are set as learnable parameters of the CNN.

In this case the neural network is a composition of linear transformations which can be written as a single linear transformation, limiting the representational power of the network.

To increase the representational power of neural networks, non-linear activation functions are used. Early works used sigmoid $\left(f(x) = \frac{1}{1+\exp(-x)}\right)$ or hyperbolic tangent $\left(f(x) = \frac{1-\exp(-2x)}{1+\exp(-2x)}\right)$ as an activation function which would squash the real valued input into a particular range, see Figure 4.6. These non-linearities are saturating in nature and learning deep neural nets with them is problematic because of gradients saturating to zero for most of the input range.

To address this issue, Nair and Hinton [2010] introduced a non-saturating non-linearity commonly known as rectified linear units (ReLU), which is one of the most widely used activation function. It computes the function $f(x) = \max(0, x)$, which in simple words, clips the activations below zero, see Figure 4.6. ReLU was found to accelerate the convergence of stochastic gradient descent for learning deep neural networks [Krizhevsky et al., 2012] and is one of the key factors for the recent success of deep neural networks. It is argued that the acceleration observed during learning is due to its non-saturating form.

4.2.2 CNN architectures for classification

Image classification involves predicting the class of the object present in an image. In this section we present some recent CNN architectures proposed for performing image classification.

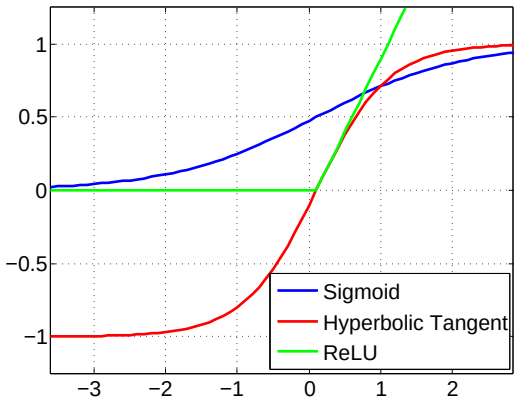


Figure 4.6 – Commonly used activation functions in neural networks: sigmoid, hyperbolic tangent and ReLU.

Krizhevsky et al. [2012] trained a deep CNN architecture to obtain state of the art results on the ImageNet classification benchmark. They outperformed the second runner up with a significant margin (16 % top-5 error rate versus 26 % error rate), and were the first to popularize the use of CNN in computer vision.

Their architecture consists of eight learnable layers, with five of them being convolution and three being fully-connected, see Figure 4.7 (top). The filter size is set to $11 \times 11, 5 \times 5, 3 \times 3, 3 \times 3, 3 \times 3$ for the five convolution layers respectively. Being unable to store the model parameters on a single GPU due to memory constraints, they propose to split the filters of a given convolution layer in two sets across two GPUs. Further, the filters for the second, fourth and fifth convolution layer take as an input only those preceding feature maps, which reside on the same GPU. Doing so reduces the number of learnable parameters. Interestingly, the filters learned on two different GPUs differ qualitatively as shown in Figure 4.7 (bottom). One set of filters is color agnostic while the other set is color specific.

Following the work of Nair and Hinton [2010], they advocate the use of Rectified linear unit (ReLU) as a non-linearity. They show that in comparison to a saturating non-linearity like hyperbolic tangent or sigmoid, ReLU leads to a faster convergence, and hence allowed them to train deeper nets. The architecture is commonly used under the name of AlexNet in the community, and lead to the widespread use of ReLU as an activation function for training deep CNN.

Simonyan and Zisserman [2015] carry out a thorough evaluation of six deep CNN architectures of increasing depth. They train networks upto a depth of 19

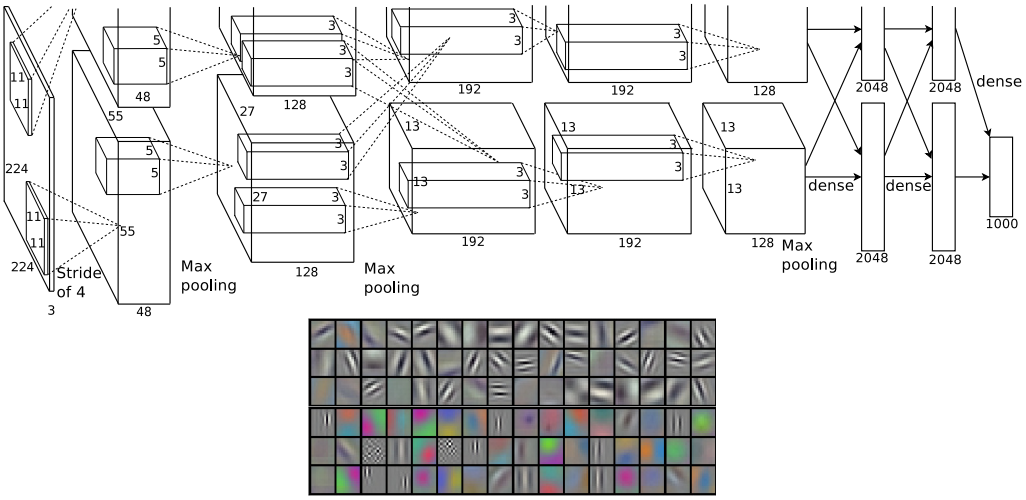


Figure 4.7 – On the top, an illustration of architecture used in AlexNet [Krizhevsky et al., 2012]. The filters of convolution layers are split in two sets across the two GPU's and the GPU's communicate only at certain layers. At the bottom, two set of 11×11 filters (top and bottom) learned for the first convolution layer. Image source [Krizhevsky et al., 2012].

layers, and achieve state of the art performance on ImageNet classification and localization tasks. In their architecture, they set the filter size of the convolution filters to be 3×3, thereby removing the filter size as a hyperparameter. This is in contrast to the standard practice where filters with a large filter size was used in the first layer (*e.g.* 11×11 with stride 4 in [Krizhevsky et al., 2012], or 7×7 with stride 2 in [Zeiler and Fergus, 2014, Sermanet et al., 2014, Szegedy et al., 2015a]). They make the following observation: A set of three convolution layers with filters of size 3×3 has an effective receptive field size of 7×7. Therefore a convolution layer with a filter size of 7×7 can be approximated by stacking three convolution layers with filter size 3×3. The advantages of their proposed stacked convolution layers are:

1. They incorporate three non-linear functions in the latter as compared to the former, and hence the learned output function would have a higher representational power.
2. Assuming that the input and output feature maps are fixed to be C, a single 7×7 convolution layer would require 7^2C^2 parameters. At the same time, the three layer convolutional stack would require $3(3^2C^2)$ parameters. The latter parameterization uses fewer parameters and can be seen as an explicit form of regularization.

In contrast to their approximation, in our work we show that the using a

3×3 filter size throughout the trellis, with sufficient number of channels, we can implement a filter of any desired filter size.

They use ReLU activation function as a non-linearity in their network, and use five Max-pooling layers interleaved between the convolution layers to decrease the spatial resolution of the filter maps. At the bottom of the network there are three fully connected layers. Out of the six different architectures evaluated, the most commonly used architecture is VGG16 which has around 138 million parameters (roughly twice when compared to AlexNet). 90% of the total number of parameters belong to the fully connected layers.

Independently to the work of Simonyan and Zisserman [2015], Szegedy et al. [2015a] developed their deep CNN architecture coined as GoogLeNet. GoogLeNet is similar to VGG16Net in the sense that it is a deep architecture (22 layers). GoogLeNet employs 7×7 convolutions with stride 2 in the first convolution layer to aggressively reduce the spatial resolution of feature maps, and hence the computational overhead.

A difference from the previous line of work is the network topology of GoogLeNet. Inspired from the topology used by Lin et al. [2014a], they introduce inception modules which collect the output responses of different filter sizes, and replace the standard convolution filter with it. To keep the computational and parameter complexity low, they propose to use 1×1 convolutions for compressing the number of input feature maps. Doing so, they are able to make the networks wider without paying a significant penalty. An illustration of inception module is shown in Figure 4.8.

To eliminate a large amount of learnable parameters they use average pooling at the top of the CNN instead of a fully connected layer like VGG16 and AlexNet. Compared to 68M and 138 M, the parameters of AlexNet and VGG16Net, GoogLeNet has less than 6M parameters and achieved state of the art result for image classifciation and detection on ILSVRC-2014 challenge. Szegedy et al. [2015b] further reduce the the computational cost and the parameter overhead by factorizing the 5×5 convolutions as a series of 3×3 convolutions. Interestingly, they also factorize the 3×3 convolution as a series of 1×3 and 3×1 asymmetric convolutions. The asymmetric factorization improved the results when used in deeper layers of the architecture.

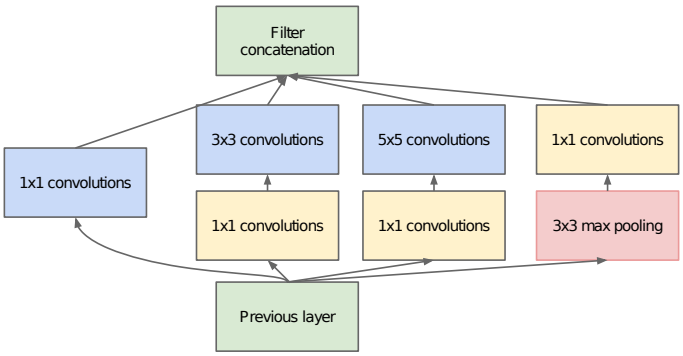


Figure 4.8 – Inception module used in GoogLeNet [Szegedy et al., 2015a]. The 1×1 convolutions are used for reducing the number of input feature maps. This is done for reducing the computational and parameter overhead of the modules. Image source [Szegedy et al., 2015a].

4.2.3 CNN architecture for segmentation

In the recent years, convolutional neural networks have made significant advances in the field of image classification. They have not only improved in task of image classification, but also related tasks such as local correspondences [Long et al., 2014, Fischer et al., 2014], object localization [Sermanet et al., 2014, Girshick et al., 2014, He et al., 2014], *etc*. Compared to these tasks, semantic segmentation involves making predictions at a finer scale. Semantic segmentation is similar to classification in the sense that it involves classifying each pixel of an input image to one of the K labels. Below, we present some recent work on CNN architectures proposed for the task of semantic segmentation.

Many of the (de)convolutional neural networks used for semantic segmentation, as well as other structured prediction tasks such as pose estimation [Pfister et al., 2015], are based on the CNN architectures developed for image classification of [Krizhevsky et al., 2012, Simonyan and Zisserman, 2015], see *e.g*. [Chen et al., 2015, Hong et al., 2015, Lin et al., 2016, Long et al., 2015, Noh et al., 2015, Tsogkas et al., 2015, Zheng et al., 2015]. In addition to convolution and pooling operators, deconvolutional networks also involve upsampling operators to increase the resolution of the predictions in the output layer [Hong et al., 2015, Noh et al., 2015]. The (de)convolutional neural networks are trained to predict label for each pixel of the input image. More formally, objective function of the network can be written as:

$$\mathcal{L}(\theta) = \sum_{p} E(X_\theta(p), l(p)), \tag{4.1}$$

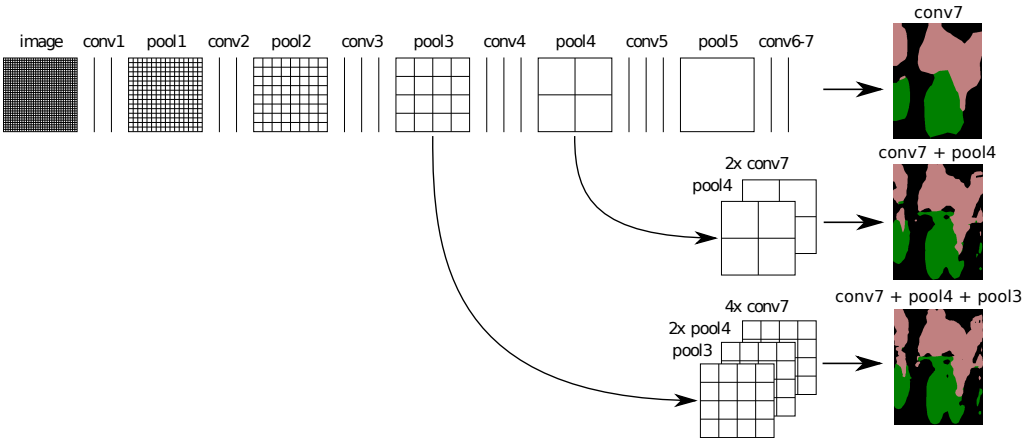


Figure 4.9 – Illustration of the skip architecture proposed in the FCN architecture [Long et al., 2015]. The network topology corresponds to a DAG and learns to combine coarse, high layer (*e.g.* conv7) information with fine, low layer (*e.g.* pool3) information. On right, qualitative results demonstrating the improvement brought in by fusing information from different layers with skip connections. Image adapted from Long et al. [2015].

where p is the pixel index, $X_\theta(p)$ is the probability distribution over the labels, predicted by a fully convolutional network with parameters θ, and $l(p)$ is the ground truth label at the pixel. $E(X_\theta(p), l(p))$ denotes per pixel multinomial logistic loss. The parameters θ are learned with standard SGD.

The work of Long et al. [2015] was an important work in the context of deep learning based image segmentation. In contrast to the previous approaches which used patch wise training [Ning et al., 2005, Pinheiro and Collobert, 2014], pre- and post-processing [Gupta et al., 2014, Hariharan et al., 2014, 2015, Farabet et al., 2013], their model is more efficient and trained end-to-end. They are the first work which interprets the classification nets as fully convolutional (see Section 4.2.1), and transfer the recent success of CNNs in classification [Krizhevsky et al., 2012, Simonyan and Zisserman, 2015, Szegedy et al., 2015a] to semantic segmentation by fine-tuning from their learned representations.

In their proposed architecture they interpolate the output response map of the penultimate layer back to the original image size with a single upsampling layer and obtain dense predictions at every pixel. They adapt the contemporary classification networks (AlexNet [Krizhevsky et al., 2012], GoogLeNet [Szegedy et al., 2015a] and VGG16Net [Simonyan and Zisserman, 2015]) into a fully convolutional networks (FCN) and transfer their learned representations by fine

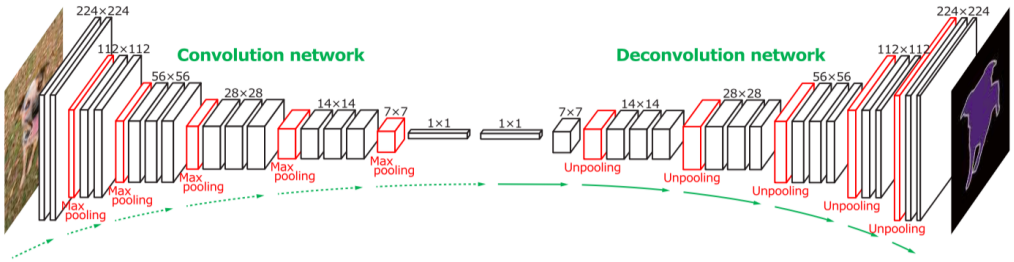


Figure 4.10 – On left, the convolution-deconvolution network proposed in [Noh et al., 2015]. The convolution network is based on the VGG16Net [Simonyan and Zisserman, 2015]. The deconvolution network constructs dense pixel-wise predictions through a series of convolution, unpooling and rectification operations. Image source [Noh et al., 2015].

tuning the network to semantic segmentation task. Even though GoogLeNet and VGG16Net attain similar classification accuracy on ImageNet, Long et al. [2015] obtain significantly inferior results with GoogLeNet for semantic segmentation.

In their experiments they observe that predictions made by penultimate layer is coarse in nature. To address this shortcoming, they add skip connections to combine predictions made from penultimate layer with predictions made from lower layers, converting the line topology to a direct acyclic graph (DAG). See Figure 4.9 for a schematic of their network architecture. Using the VGG16Net equipped with the skip connections, they obtain state-of-the-art results on different semantic segmentation datasets.

The work of Noh et al. [2015] builds upon the work of Long et al. [2015] by learning a deep deconvolutional network. Their architecture is composed of two parts: a convolutional and a deconvolutional network. The convolution network represents a feature extractor which transforms an image into a multi-dimensional feature representation via a series of convolution and pooling layers. The deconvolution network takes the feature representation as an input and outputs an image segmentation. The deconvolutional network is composed of convolution and unpooling layers, which gradually upsample the response map so as to obtain the predictions of the same size as the input image. They advocate the use of unpooling layers [Zeiler and Fergus, 2014, Zeiler et al., 2011], which perform unpooling operation by using the pooled location from the corresponding pooling layers of the convolution network. Figure 4.10 illustrates the proposed architecture composed with the convolution and the deconvolution network. To

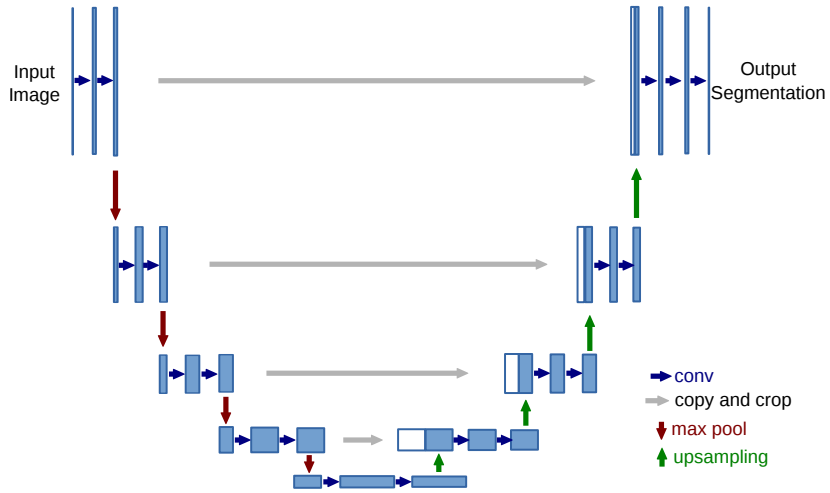


Figure 4.11 – Schematic illustration of U-Net architecture [Ronneberger et al., 2015]. Distant convolution and deconvolution layers are linked with additional links. Blue and white rectangles indicate output of a convolution operation and copied channels respectively. The thickness of rectangles is propotional to number of channels. Image adapted from [Ronneberger et al., 2015].

obtain invariance from scale, they apply their trained architecture to individual object proposals and obtain instance-wise segmentations. In the post-processing step, they combine the instance-wise segmentations to output the final semantic segmentation. Using their proposed architecture and post-processing, they obtain a significant gain over the work of Long et al. [2015].

Independently to the work of Noh et al. [2015], Ronneberger et al. [2015] propose a similar architecture (U-Net) but for the task of biomedical image segmentation. U-Net architecture also involves a convolution network and deconvolution network, but couples the distant convolutional and deconvolutional layers of the same resolution with additional links, see Figure 4.11. The additional links are added in order to preserve the fine grained details, which would otherwise be lost in the convolution network. Milletari et al. [2016] builds upon this work for the task of volumetric image segmentation. They use an architecture similar to that of [Ronneberger et al., 2015], but replace the 2D convolution filter with a 3D convolution filter.

Multi-scale architectures are commonly used for semantic segmentation [Chen et al., 2015, Farabet et al., 2013, Lin et al., 2016]. They re-sample the input image to several resolutions, process them independently for several layers, and then fuse all streams by upsampling to the maximum resolution. The result is

passed through several more layers to produce the final output. Such architectures can be thought of as a tree that has leafs in different re-scaled input images, and the root at the output.

In Section 4.2.2 and Section 4.2.3 we have presented a brief overview of different CNN architectures used for image classification and segmentation respectively. Hand-crafting the CNN architecture plays a crucial role in the recent success of CNNs. The networks differ in terms of different hyperparameters of CNN architecture, such as depth, number of filters, number of convolution and pooling layers, connecting links, *etc*. Finding the right architecture for each task is not trivial, and remains a manual and a tedious process. In our work we show that nearly all of the (de)convolutional networks discussed above are embedded in our fabric, either as paths or other simple sub-graphs. Before describing the construction of fabric in Section 4.3, we give a brief review of work closely related to our work presented in this chapter.

4.2.4 Closely related work

In this section we give an overview of recent work on CNN architectures closely related to our work. Some of them share motivation to simplify CNN architecture selection problem, whereas some share resemblance to our work in terms of the CNN architecture. We highlight the differences we observe in comparison to the work presented in this chapter.

The work of Zhou et al. [2015] is closely related to ours. They interlink CNN models that take input from re-scaled versions of the input image. The structure of their model is related to our trellis, but lacks a sparse connectivity pattern across the channel dimension. Moreover, they set the number of channels, filter size per layer and scale in an ad-hoc manner. They explore their networks only for semantic segmentation, not for image classification. Finally, they did not observe that structures similar to theirs suffice to span a vast class of (de)convolutional networks, which is our main result.

Misra et al. [2016] propose related cross-stitch networks that exchange information across corresponding layers of two copies of the same architecture that produces two different outputs. Their approach is based on the architecture of Krizhevsky et al. [2012], and does not address the network architecture selection problem.

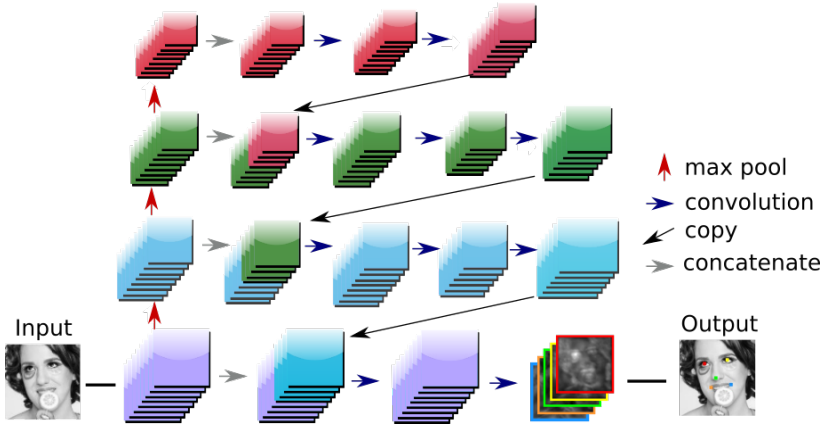


Figure 4.12 – Architecture of Recombinator networks [Honari et al., 2016] for landmark localization. Each branch, takes upsampled feature maps from the coarser branch.

Kulkarni et al. [2015] use ℓ_1 regularization to automatically select the number of units in "fully-connected" layers of CNN architectures for classification. Although their approach does not extend to determine more general architectural design choices, it might be possible to use such regularization techniques to select the number of channels and/or layers of our fabrics.

Springenberg et al. [2015] experimentally observed that the use of max-pooling in CNN architectures is not always beneficial as opposed to only using strided convolutions. In our work we go one step further and show that ReLU units and (strided) convolutions suffice to implement max-pooling operators in our trellis. Their work, similar to ours, also strives to simplify architecture design. Our results, however, reach much further than only removing one pooling operator from the architectural hyperparameters.

Lee et al. [2016] generalize the max and average operators by computing both max and average pooling, and then fusing the result in a possibly data-driven manner. Our fabrics also generalize max and average pooling, but instead of adding more elementary operators, we show that settings weights in a network with fewer elementary operators is enough for this generalization.

Honari et al. [2016] argue that coarse and fine response maps preserve discriminative and localization details respectively, and propose recombinator network architecture to fuse information at different spatial resolutions. Traditional multi-resolution approaches [Long et al., 2015, Hariharan et al., 2015] take a

weighted sum of upsampled feature maps. In contrast, they upsample the output at a given scale, and concatenate it with the input feature maps at a finer scale, see Figure 4.12. There work, similar to ours, fuses information at mutliple scales, however, our work goes beyond that by automating CNN architecture selection problem.

Dropout [Srivastava et al., 2014] and swapout [Singh et al., 2016] are stochastic training methods related to our work. They can be understood as approximately averaging over an exponential number of variations of a given architecture. Our approach, on the other hand, allows to leverage an exponentially large class of architectures (ordering of pooling and convolutional layers, type of pooling operator, *etc*.) by means of continuous optimization. Note that these approaches are orthogonal and can be applied to fabrics.

While multi-dimensional networks have been proposed in the past, *e.g*. to process non-sequential data with recurrent nets [Graves et al., 2007, Kalchbrenner et al., 2016], to the best of our knowledge they have not been explored as a "basis" to span large classes of (de)convolutional neural networks.

4.3 The fabric of convolutional neural networks

We now give a precise definition of our trellis model, and show in Section 4.3.2 that most architectural design choices of (de)convolutional nets become irrelevant for sufficiently large trellises. Finally, we analyze the number of response maps, parameters, and activations of our trellises in Section 4.3.3.

4.3.1 Weaving the convolutional neural fabric

Each node in the trellis represents a response map with the same dimension as the input signal (1D for audio, 2D for images, 3D for video). The structure of the trellis over the nodes is spanned by three axes. A **l**ayer axis along which all edges advance, which rules out any cycles, and which is analogous to the depth axis of a CNN. A **s**cale axis along which response maps of different resolutions are organized from fine to coarse, neighboring resolutions are separated by a factor two. A **c**hannel axis along which different response maps of the same scale and layer are organized. In practice we use S scales when we process inputs of size 2^{S-1}, *e.g*. for 32×32 images we use six scales, so as to obtain a full scale pyramid from the input resolution all the way to the coarsest 1×1 activations.

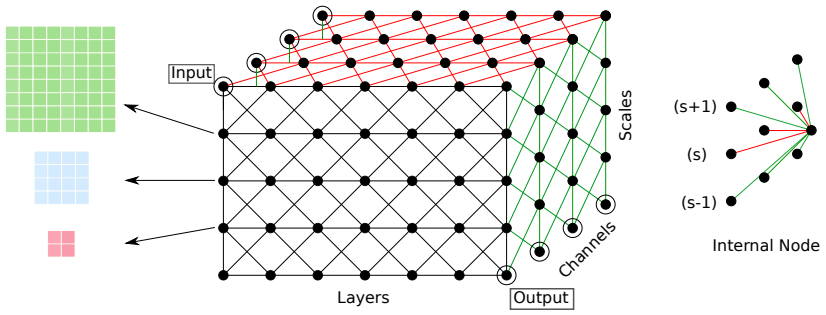


Figure 4.13 – On the left, schematic illustration of the sparse homogeneous edge structure in the trellis. Channel connectivity pattern at the same scale (red) and across finer and coarser scale (green) depict sparse connect structure. See text for more details. On the right, connectivity of a channel of an internal node to channels of preceding nodes is shown. The internal node is connected to 3 channels at the same scale (red), 3 at the finer and 3 at coarser scales (green).

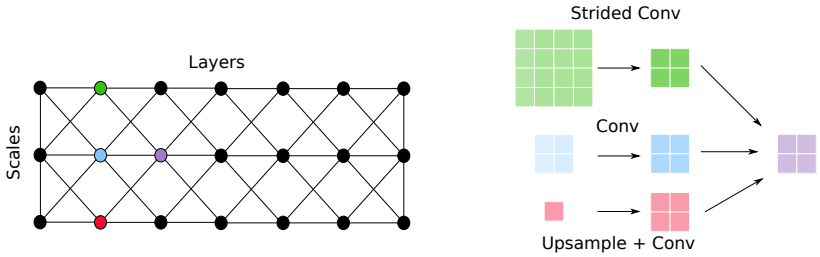


Figure 4.14 – Schematic illustration of how a node (purple) in a trellis combines the input from different scales. Input from a finer scale is obtained with a stride-2 convolution (green), and from a coarser scale by upsampling the feature map, followed by a convolution (red). Input at the same scale is obtained with stride-1 convolution which preserves the spatial resolution (blue). The output feature map (purple) is generated by summing the three response maps.

We now define a sparse and homogeneous edge structure. Each node is connected to a 3×3 scale–channel neighborhood in the previous layer, *i.e*. channel c at scale s receives input from channels $\{c-1, c, c+1\}$ at scales $\{s-1, s, s+1\}$ in the previous layer, see Figure 4.13. Input from a finer scale is obtained via strided convolution, and input from a coarser scale by convolution, after upsampling by padding zeros around the activations at the coarser level, see Figure 4.14.

Activations in the trellis are thus a linear function over multi-dimensional neighborhoods, *i.e*. a four dimensional $3\times3\times3\times3$ neighborhood when using 2D input images. The propagation is, however, only convolutional across the input dimensions, and not across the scale and layer axes. Note that "fully connected"

layers of a CNN correspond to nodes and edges along the coarsest 1×1 scale of the trellis. Rectified linear units (ReLUs) are used at all nodes. Figure 4.14 (left) illustrates the connectivity pattern in 2D, omitting the channel dimension for clarity.

All channels in the first layer at finest resolution are connected to all channels in the input signal. The first layer contains additional edges to distribute the signal across coarser scales, see the vertical edges in Figure 4.14 (left). More precisely, within the first layer, channel c at scale s receives input from channels $\{c-1, c, c+1\}$ from the previous scale $s-1$. Similarly, edges within the last layer collect the signal towards the output. Note that these additional edges do not create any cycles in the trellis, and that the edge-structure within the first and last layer is reminiscent of the 2D trellis in Figure 4.14.

4.3.2 Stitching chain-structured networks on the fabric

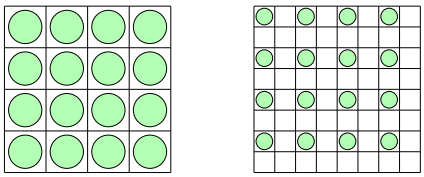
We now explicitly show how various architectures are embedded in the trellis, demonstrating it subsumes essentially all (de)conv. models discussed above. In practice, learning configures the trellis to behave as one architecture or another, but generally as an ensemble of all embedded architectures.

For all but the last of the following paragraphs, it is sufficient to consider a 2D trellis, as in Figure 4.14, where each node contains the response maps of C channels with dense connectivity among channels.

Re-sampling operators Several operators are used in (de)convolutional networks to change the resolution of response maps, strided convolution is the most basic one. Stride-two convolutions are used in the trellis on all fine-to-coarse edges, larger strides can be obtained by following multiple such edges.

Average pooling across small regions is also strided convolution with uniform filter weights, and thus available in the trellis. See the next paragraph for averaging pooling over larger areas.

Consider *max-pooling* over a 2×2 region, larger sizes are obtained by repetition. Let a and b represent the values of two vertically neighboring pixels. Use one layer and three channels to compute $(a+b)/2$, $(a-b)/2$, and $(b-a)/2)$. After ReLU, at most only two terms remain non-zero. A second layer can compute the sum of the three terms, which equals $\max(a, b)$. Each pixel now contains the maximum of its value and that of its vertical neighbor. Repeating the same in



$$\frac{1}{4}\begin{pmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{pmatrix} \qquad \begin{pmatrix}1 & 1 & 0\\ 1 & 1 & 0\\ 0 & 0 & 0\end{pmatrix}$$

Bilinear Nearest neighbor

Figure 4.15 – Illustration of how fabrics can implement different upsampling operators. In the top, fabrics upsample a feature map by zero padding, followed by a convolution. In the bottom, setting the convolution kernel to specific filters implements bi-linear and nearest neighbor interpolation.

the horizontal direction, and sub-sampling by a factor two, gives the output of 2×2 max-pooling. This process can also be adapted to show that a network of *MaxOut units* [Goodfellow et al., 2013] can be implemented in a trellis of ReLU units.

Bi-linear interpolation is commonly used in deconvolutional networks. Factor-two interpolation can be represented in our trellis on coarse-to-fine edges by using a filter that has 1 in the center, $1/4$ on corners, and $1/2$ elsewhere. Interpolation by larger powers of two can be obtained by repetition. Similarly, *Nearest neighbor interpolation* is obtained using a filter that is 1 in the four top left entries and zero elsewhere, see Figure 4.15.

Filter sizes To implement a 5×5 filter we first compute nine intermediate channels to obtain a vectorized version of the 3×3 neighborhood at each pixel, using filters that contain a single one, and are zero elsewhere. A second 3×3 convolution can then aggregate values across the original 5×5 patch, and output the desired convolution, see Figure 4.16. Any 5×5 filter can be implemented in this way, not only factorized approximations, *c.f.* Simonyan and Zisserman [2015]. With an appropriate number of channels in the trellis, repetition allows to implement every filter of any desired size.

Ordering convolution and re-sampling As shown in Figure 4.1, chain-structured (de)convolutional nets correspond to paths in the trellis. If weights on edges out-

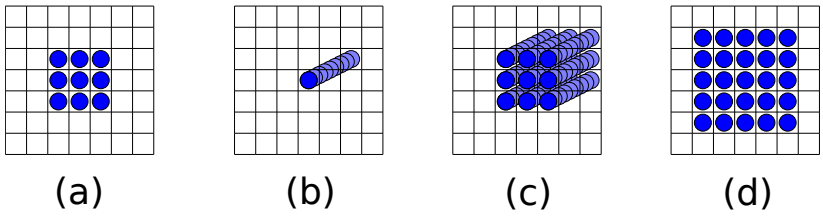


Figure 4.16 – In this illustration we consider computing a 5×5 convolution over a single channel with 3×3 convolutions in the fabric. The first convolution operation computes 9 intermediate channels (b) to obtain a vectorized version of 3×3 neighborhood in the input (a). Performing a 3×3 convolution over the intermediate channels (b) allows the filter to access the 5×5 patch in the input (d).

side a path are set to zero, a chain-structured (de)convolutional net with a particular sequencing of convolutions and re-sampling operators is obtained. A trellis that spans $S+1$ scales and $L+1$ layers contains more than $\binom{L}{S}$ chain-structured CNNs, since this corresponds to the number of ways to spread S sub-sampling operators across the L steps to go from the first to the last layer. More CNNs are embedded, *e.g*. by exploiting edges within the first and last layer, or by including intermediate up-sampling operators.

Networks beyond chain-structured ones, see *e.g*. Farabet et al. [2013], Long et al. [2015], Ronneberger et al. [2015], are also embedded in the trellis, by activating a larger subset of edges than a single path, *e.g*. a tree structure for the multi-scale net of Farabet et al. [2013].

Channel connectivity pattern Although most networks in the literature use dense connectivity across channels between successive layers, this is not a necessity. For example, Krizhevsky et al. [2012] used a network that is partially split across two independent processing streams.

In Figure 4.17 we demonstrate that our trellis, which is sparsely connected along the channel axis, suffices to emulate densely connected convolution layers. This is achieved by copying channels, convolving them, and then locally aggregating them. Both the copy and sum process are based on local channel interactions and convolutions with filters that are either entirely zero, or identity filters which are all zero except for a single 1 in the center. While more efficient constructions exist to represent the densely connected layer in our trellis, the one presented here is simple to understand and suffices to demonstrate feasibility. Note that in practice learning automatically configures the trellis.

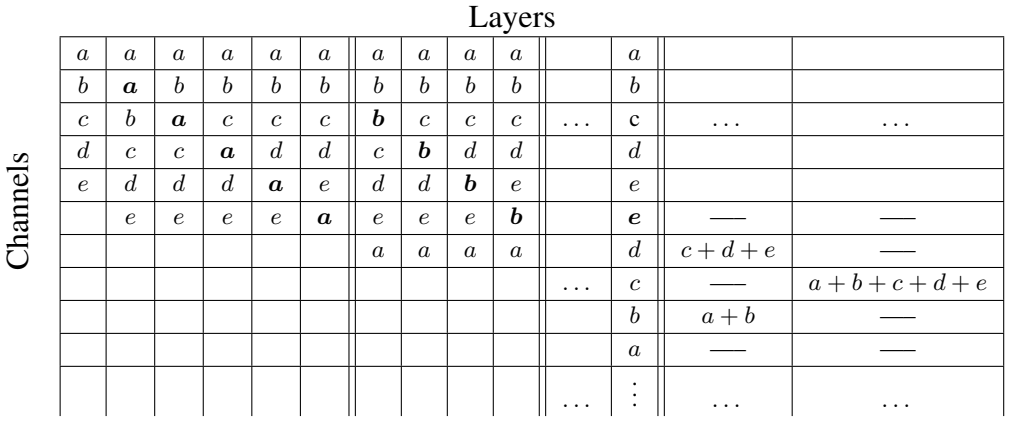


Figure 4.17 – Schematic representation of a dense-channel-connect layer in our sparse trellis using local copy and swap operations. The five input channels $a, \ldots, e$ are first copied; more copies are generated by repetition. Channels are then convolved and locally aggregated in the last two layers to compute the desired output. Channels in rows, layers in columns, scales are ignored for simplicity.

Table 4.1 – Number of response maps, parameters, activations, and multiplications for a trellis with L layers, S scales, C channels, for $2D$ inputs of size $N \times N$ pixels. Channel doubling across scales used in the bottom row.

# chan. / scale	# resp. maps	# parameters (sparse)	# parameters (dense)	# activations	# multiplications (dense)	# multiplications (sparse)
constant	$C \cdot L \cdot S$	$C \cdot L \cdot 3^3 \cdot 3 \cdot S$	$C \cdot L \cdot 3^3 \cdot C \cdot S$	$C \cdot L \cdot N^2 \cdot \frac{4}{3}$	$C \cdot L \cdot N^2 \cdot 27C$	$C \cdot L \cdot N^2 \cdot 81$
doubling	$C \cdot L \cdot 2^S$	$C \cdot L \cdot 3^3 \cdot 3 \cdot 2^S$	$C \cdot L \cdot 3^3 \cdot C \cdot 4^S \cdot \frac{4}{9}$	$C \cdot L \cdot N^2 \cdot 2$	$C \cdot L \cdot N^2 \cdot 25\frac{9}{8}C \cdot S$	$C \cdot L \cdot N^2 \cdot 135$

Both the copy and sum process generally require more than one trellis layer to execute. In the copying process, intermediate ReLUs do not affect the result since the copied values themselves are non-negative outputs of ReLUs. In the convolve-and-sum process care has to be taken since one convolution might give negative outputs, even if the sum of convolutions is positive. To handle this correctly, it suffices to shift the activations by subtracting from the bias of every convolution i the minimum possible corresponding output $a_i^{\min}$ (which exists for any bounded input domain). Using the adjusted bias, the output of the convolution is now guaranteed to be non-negative, and to propagate properly in the copy and sum process. In the last step of summing the convolved channels, we can add back $\sum_i a_i^{\min}$ to shift the activations back to recover the desired sum of convolved channels.

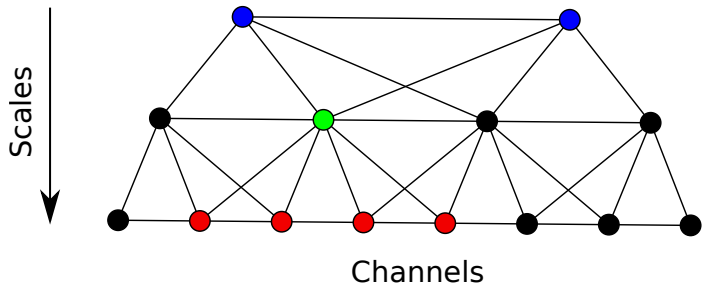


Figure 4.18 – Diagram of sparse channel connectivity from one layer to another in the channel-doubling trellis. Channels are laid out horizontally, scales vertically. Each internal node (green), channel, is connected to nine other nodes at the previous layer: four channels (red) at a coarser resolution, two (blue) at a finer resolution, and to itself and neighboring channels at the same resolution.

4.3.3 Analysis of the number of parameters and activations

For our analysis we ignore border effects, and consider every node to be an internal one. In the top row of Table 4.1 we state the total number of response maps throughout the trellis, and the number of parameters when channels are sparsely or densely connected. We also state the number of activations in the trellis, which determines the memory usage of back-propagation during learning.

While embedding an exponential number of architectures in the number of layers L and channels C, the number of activations and thus the memory cost during learning grows only linearly in C and L.

The number of parameters is linear in the number of scales S. For sparsely connected channels, the number of parameters grows also linearly with the number of channels C , while it grows quadratically with C in case of dense connectivity. The number of activations grows exponentially with the number of scales, since each scale layer doubles the resolution response maps. In other words, the number of scales is logarithmic in the input size, *e.g*. six for 32×32 images, and nine for 256×256 images.

As an example, the largest models we trained for 32×32 input have $L = 16$ layers and $C = 256$ channels, resulting in 2M parameters (255M for dense), and 6M activations. For 256×256 input we used upto $L = 16$ layers and $C = 64$ channels, resulting in 0.7 M parameters (15M for dense), and 89M activations. For reference, the VGG-19 model has 144M parameters and 14M activations.

In addition to the trellis structure defined above, we analyze a second trellis in the second row of Table 4.1; here the number of channels doubles when moving one scale coarser instead of being constant. In the case of sparsely connected channels, we adapt the local connectivity pattern between nodes to accommodate for the varying number channels per scale, see Figure 4.18 for an illustration. Each node still connects to nine other nodes at the previous layer. Whereas before a node at scale s took input from three channels from scales $\{s-1, s, s+1\}$, it now takes two inputs from scale $s-1$, three from scale s, and four from scale $s+1$.

This results in a number of channels throughout the scale pyramid that is exponential in the number of scales S for a given number of "base channels" at the finest resolution C. For 32×32 input images the total number of channels is roughly $11\times$ larger, while for 256×256 images we get roughly 57 more channels. The last column of Table 4.1 shows that the number of activations, however, grows only by 50%. Since each node still takes nine inputs from the previous layer, the computational cost grows also only by 50%. In case of sparse channel connection, the number of parameters grows by the same factor $2^S/S$. In case of dense connections, however, the number of parameters explodes with a factor $\frac{4}{9}4^S/S$. That is, roughly a factor 303 for 32×32 input, and 12,945 for 256×256 input. Therefore, the channel-doubling variant of our trellis is very attractive, provided that sparse channel connectivity is used. Similar channel-doubling is also used in the well-known VGG-16/19 architectures [Simonyan and Zisserman, 2015].

4.4 Experimental evaluation

In this section we detail the experimental evaluation of dense and sparse trellis for the task of image segmentation and classification. First we present the datasets and evaluation protocols used in our experiments in Section 4.4.1. Next, we provide quantitative evaluation of the dense and sparse trellises and compare our results with relevant state-of-the-art methods. Finally in Section 4.4.6 we visualize the structure of the learned trellises on different datasets, and observe qualitative differences between them.

4.4.1 Datasets and experimental protocol

Part Labels dataset This dataset [Kae et al., 2013a] consists of 2,927 face images from the LFW dataset [Huang et al., 2007], with pixel-level annotations into the classes *hair*, *skin*, and *background*. We use the standard evaluation protocol

which specifies training, validation and test sets of 1,500, 500 and 927 images, respectively. We report performance in terms of pixel-level and superpixel-level accuracy. For the latter, we average the class probabilities over pixels contained in the superpixel. To augment the train set we add flipped versions of the images, and do not use any other data augmentation techniques. The batch size is set to 64, and we train our networks upto 8k iterations. The learning rate is dropped by factor 10 after 4k and 6k iterations.

MNIST This dataset [LeCun et al., 1998] consists of 28×28 pixel images of the handwritten digits $\{0, \ldots, 9\}$. We use the standard split of the dataset into 50k training samples, 10k validation samples and 10k test samples. Pixel values are normalized to $[0, 1]$ by dividing them by 255. We augment the train data by randomly positioning the original image on a 32×32 pixel canvas. The batch size is set to 64, and we train our networks upto 20k iterations. The learning rate is dropped by factor 10 after every 5k iterations.

CIFAR10 The CIFAR-10 dataset [1] [Krizhevsky, 2009] consists of 50k 32×32 training images and 10k testing images in 10 classes. We hold out 5k training images as validation set, and use the remaining 45k as the training set. To augment the data, we follow common practice, see *e.g.* [Lin et al., 2013, Goodfellow et al., 2013], and pad the images with zeros to a 40×40 image and then take a random 32×32 crop, in addition we add flipped versions of these images. The batch size is set to 128, and we train our networks upto 40k iterations. The learning rate is dropped by factor 10 after 20k and 30k iterations.

Training protocol We train our trellises using SGD with momentum of 0.9. We set the initial learning rate to 0.1 in all experiments. After each node in the trellis we apply batch normalization [Ioffe and Szegedy, 2015], and regularize the model with weight decay of 10^{-4}. We do not apply dropout [Srivastava et al., 2014] at any node in the trellis.

For all the three datasets we train dense and sparse trellises with various numbers of channels and layers to find the best architecutre (in terms of channels and layers). In these experiments, we train the models only on the train set, and report results on the test set. To compare our method to state-of-the-art, we train the previously found best architecture on both the validation and the train set. In both set of experiments, the validation set is used to determine the optimal number of training epochs. For the experiments reported here, we use a constant number of channels per scale.

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

Table 4.2 – Superpixel-level accuracy on Part Labels for CNF-dense. Number of parameters given in parentheses.

Layers / Channels	4		16		64	
2	93.57	*(8K)*	95.26	*(124K)*	95.33	*(2M)*
4	93.67	*(16K)*	95.05	*(249K)*	95.20	*(4M)*
8	95.09	*(31K)*	95.22	*(498K)*	**95.39**	*(8M)*
16	94.92	*(62K)*	95.29	*(995K)*	95.34	*(16M)*

Table 4.3 – Superpixel-level accuracy on Part Labels for CNF-sparse. Number of parameters given in parentheses.

Layers / Channels	4		16		64	
2	91.95	*(6K)*	94.76	*(23K)*	95.14	*(93K)*
4	93.94	*(12K)*	95.02	*(47K)*	95.34	*(187K)*
8	94.87	*(23K)*	**95.48**	*(93K)*	95.46	*(373K)*
16	95.15	*(47K)*	95.38	*(187K)*	95.26	*(746K)*

4.4.2 Evaluation on Part Labels

On this dataset we evaluate the performance of fabrics for the task of image segmentation. The performance is measured in terms of super-pixel level labelling accuracy. In Table 4.2 and Table 4.3 we evaluate dense and sparse trellis by varying the number of layers and channels. Besides the labelling acuracy, we also report the number of parameters used for each model. The best performance for dense and sparse trellis is obtained with 8 layers, 64 channels and 8 layers,16 channels respectively. The latter uses one order of magnitude less parameters than the former, and obtains slightly better performance. As it can be seen from the results in the table, performance of larger trellises is better than that of smaller ones.

In Table 4.4 we compare our results with the state-of-the-art, both in terms of accuracy and the number of parameters. For accuracy, we report both the super-pixel level accuracy and pixel level accuracy. We obtained a super-pixel accuracy of 95.6 using both sparse and dense trellises. Our sparse convolutional neural fabric (CNF) results improve over Kae et al. [2013a] which also employs CRF and RBM shape models. In contrast, we predict all pixels independently

Figure 4.19 – Examples form the Part Labels test set: input image (left), ground-truth labels (middle), and superpixel-level labels from our sparse CNF model with 8 layers and 16 channels (right).

Table 4.4 – Comparison of our results with the state of the art on Part Labels. The performance is reported in terms of super-pixel and pixel level accuracy.

	Year	# Params.	SP Acccur.	P Accur.
Tsogkas et al. [2015]	2015	>414 M	96.97	—
Zheng et al. [2015]	2015	>138 M	96.59	—
Liu et al. [2015]	2015	>33 M	—	95.24
Kae et al. [2013a]	2013	0.7 M	94.95	—
Ours: CNF-sparse ($L = 8, C = 16$)		0.1 M	95.58	94.60
Ours: CNF-dense ($L = 8, C = 64$)		8.0 M	95.63	94.82

using a model with about seven times less parameters. In Figure 4.19 we show two examples of predicted segmentation maps with the CNF-sparse trellis.

Our results are slightly worse than Tsogkas et al. [2015], Zheng et al. [2015]. Their results are based on VGG-16 network, which has three orders of magnitude more parameters than our sparse trellis, and has been trained from over 1M ImageNet images. In contrast, we have trained our model from scratch using only 2,000 images. Moreover, Tsogkas et al. [2015] also includes CRF and RBM for spatial regularization and train separate CNN's for each category.

4.4.3 Evaluation on MNIST

In this section, we evaluate our dense and sparse trellis for the task of image classification on MNIST dataset. In Table 4.5 and Table 4.6, we report the error

Table 4.5 – Error rate on MNIST for CNF-dense. Number of parameters given in parentheses.

Layers / Channels	4	8	16	32	64	128
2	3.47 *(8K)*	1.46 *(31K)*	0.73 *(124K)*	0.65 *(498K)*	0.59 *(2M)*	0.59 *(8M)*
4	2.74 *(16K)*	0.88 *(62K)*	0.67 *(249K)*	0.54 *(1M)*	0.59 *(4M)*	0.51 *(16M)*
8	1.65 *(31K)*	0.70 *(124K)*	0.60 *(498K)*	0.52 *(2M)*	**0.39** *(8M)*	0.46 *(32M)*
16	1.04 *(62K)*	0.60 *(249K)*	0.50 *(1M)*	0.55 *(4M)*	0.39 *(16M)*	0.47 *(64M)*
32	1.29 *(124K)*	0.92 *(498K)*	0.64 *(2M)*	0.57 *(8M)*	0.61 *(32M)*	0.56 *(128M)*

Table 4.6 – Error rate on MNIST for CNF-sparse. Number of parameters given in parentheses.

Layers / Channels	4	8	16	32	64	128
2	4.94 *(4K)*	2.32 *(8K)*	1.68 *(16K)*	1.40 *(31K)*	0.97 *(62K)*	1.00 *(124K)*
4	2.96 *(8K)*	1.69 *(16K)*	1.14 *(31K)*	0.96 *(62K)*	0.98 *(124K)*	0.78 *(249K)*
8	1.94 *(16K)*	1.12 *(31K)*	0.87 *(62K)*	0.69 *(124K)*	0.79 *(249K)*	0.57 *(498K)*
16	1.13 *(31K)*	0.91 *(62K)*	0.71 *(124K)*	**0.56** *(249K)*	0.68 *(498K)*	0.70 *(1M)*
32	1.37 *(62K)*	0.88 *(124K)*	0.67 *(249K)*	0.61 *(498K)*	0.77 *(1M)*	0.73 *(2M)*

rate and number of parameters for different configurations of dense and sparse trellises. For the dense trellis, the largest model we trained consists of 128M parameters, but performs inferior to the best result we obtain with 8 layers and 64 channels (0.56 vs 0.39). This is possibly due to over-fitting. With the sparse trellis we obtain slightly inferior results as compared to the dense trellis (0.56 vs 0.39) but with an order of magnitude less parameters.

In Table 4.7 we compare our results to recent state-of-the-art. We excluded several more accurate results reported in the literature, since they are based on significantly more elaborate data augmentation methods. We obtain error rates of 0.48% and 0.33% with sparse and dense trellises respectively. Our result with densely connected trellis is comparable to those of Kalchbrenner et al. [2016], Wan et al. [2013], which use similar data augmentation. Our sparse model, which has 20$\times$ less parameters than the dense variant, and yields an error of 0.48% which is slightly higher. In Figure 4.20 we plot the 33 errors among the 10,000 test samples for the result of densely connected trellis reported in Table 4.7. A significant portion of these errors are due to missing strokes in the hand-written digits, see *e.g.* row three, column two in the figure.

Table 4.7 – Comparison of our results with the state of the art on MNIST. Data augmentation with translation and flipping is denoted by T and F respectively, N denotes no data augmentation.

	Year	Augmentation	# Params.	Error (%)
Chang and Chen [2015]	2015	N	447K	0.24
Lee et al. [2015]	2015	N		0.31
Grid LSTM [Kalchbrenner et al., 2016]	2016	T		0.32
Dropconnect [Wan et al., 2013]	2013	T	379K	0.32
CKN [Mairal et al., 2014]	2014	N	43 K	0.39
Maxout [Goodfellow et al., 2013]	2013	N	420 K	0.45
Network in Network [Lin et al., 2013]	2013	N		0.47
Ours: CNF-sparse ($L = 16, C = 32$)		T	249 K	0.48
Ours: CNF-dense ($L = 8, C = 64$)		T	5.3 M	0.33

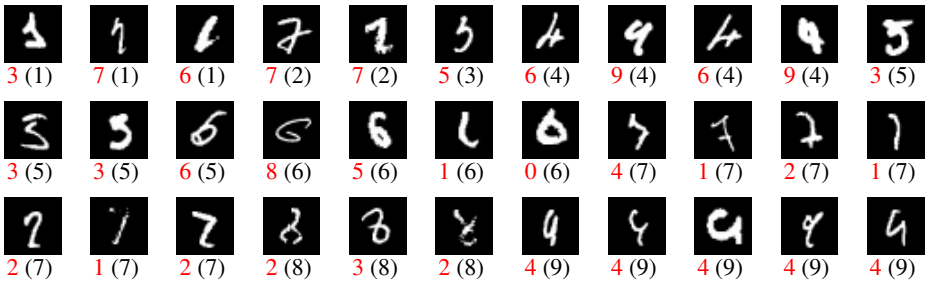


Figure 4.20 – All 33 errors among 10,000 test samples of MNIST for the result of densely connected trellis reported in Table 4.7. The prediction and groundtruth are reported in red and black respectively.

4.4.4 Evaluation on CIFAR10

In this section we evaluate the trellis on CIFAR10 dataset for the task of image classification. In Table 4.8 and Table 4.10, we report results for different configurations of dense and sparse trellis. The parameters for the different configurations are provided in Table 4.9 and Table 4.11 respectively. For CNF-dense the best results are obtained with 8 layers. 256 channels, and 127M parameters. Models with 16 layers and a large number of channels (*e.g.* 128 and 256) perform worse as compared to the relatively smaller trellises. This is possibly because of optimization difficulties we face when learning trellises with a large number of parameters and embedded architectures. On this dataset, the error rates obtained with sparse trellis is significantly worse. We are investigating the

Table 4.8 – Error rate on CIFAR10 for CNF-dense. Number of parameters given in Table 4.9.

Layers / Channels	2	4	8	16	32	64	128	256
2	68.70	50.96	33.66	23.92	18.83	15.72	13.79	13.11
4	62.34	41.92	27.76	19.22	14.65	13.38	12.09	10.06
8	58.26	35.12	22.53	15.57	13.05	10.88	9.42	**9.31**
16	50.31	28.27	19.03	13.57	10.95	9.65	10.63	14.27

Table 4.9 – Number of parameters for CIFAR10, CNF-dense.

Layers / Channels	2	4	8	16	32	64	128	256
2	*(2K)*	*(8K)*	*(31K)*	*(124K)*	*(498K)*	*(2M)*	*(8M)*	*(32M)*
4	*(4K)*	*(16K)*	*(62K)*	*(249K)*	*(1M)*	*(4M)*	*(16M)*	*(64M)*
8	*(8K)*	*(31K)*	*(124K)*	*(498K)*	*(2M)*	*(8M)*	*(32M)*	*(127M)*
16	*(16K)*	*(62K)*	*(249K)*	*(1M)*	*(4M)*	*(16M)*	*(64M)*	*(255M)*

Table 4.10 – Error rate on CIFAR10 for CNF-sparse. Number of parameters for the different configurations are provided in Table 4.11.

Layers / Channels	2	4	8	16	32	64	128	256
2	68.70	49.65	48.95	34.48	31.48	28.82	27.67	25.56
4	62.34	43.69	34.28	30.07	26.18	25.14	22.96	22.60
8	58.26	40.02	28.10	24.44	22.12	22.20	20.66	21.38
16	50.31	32.28	25.70	22.65	19.74	19.07	19.05	**18.89**

reason for this exception.

In Table 4.12 we compare our results to the state of the art. Our error rate of 7.43% with a dense trellis is competitive to that reported with MaxOut networks Goodfellow et al. [2013]. We also obtain results comparable to that reported with All Convolutional Net [Springenberg et al., 2015]. They obtain a slightly better rate, with an order of magnitude fewer parameters. This is due to the fact that their network has been hand-crafted for optimizing the performance while keeping the parameter count low. Our trellis construction enjoys the benefit of having far less hyperparameters to be tuned at the cost of having an order of magnitude more parameters.

Table 4.11 – Number of parameters for CIFAR10, CNF-sparse.

Layers / Channels	2	4	8	16	32	64	128	256
2	*(2K)*	*(4K)*	*(8K)*	*(16K)*	*(31K)*	*(62K)*	*(124K)*	*(249K)*
4	*(4K)*	*(8K)*	*(16K)*	*(31K)*	*(62K)*	*(124K)*	*(249K)*	*(498K)*
8	*(8K)*	*(16K)*	*(31K)*	*(62K)*	*(124K)*	*(249K)*	*(498K)*	*(1M)*
16	*(16K)*	*(31K)*	*(62K)*	*(124K)*	*(249K)*	*(498K)*	*(1M)*	*(2M)*

Table 4.12 – Comparison of our results with the state of the art on CIFAR10. Data augmentation with translation, flipping, scaling and rotation are denoted by T, F, S and R respectively.

	Year	Augmentation	# Params.	Error (%)
Lee et al. [2015]	2015	T+F	1.8M	6.05
Chang and Chen [2015]	2015	T+F	1.6M	6.75
All Convolutional Net [Springenberg et al., 2015]	2015	T+F	1.3 M	7.25
[Mishkin and Matas, 2016]	2016	T+F	-	7.40
Network in Network [Lin et al., 2013]	2013	T+F	1 M	8.81
Dropconnect [Wan et al., 2013]	2013	T+F+S+R	19M	9.32
MaxOut [Goodfellow et al., 2013]	2013	T+F	>6 M	9.38
Ours: CNF-sparse ($L = 16, C = 64$)		T+F	2M	18.89
Ours: CNF-dense ($L = 8, C = 128$)		T+F	32M	7.43

Mishkin and Matas [2016] obtained an error rate of 5.84% by using residual connections [He et al., 2016], MaxOut activations [Goodfellow et al., 2013], and LSUV weight initialization (which we can, but did not yet, use in our approach). Without residual connections, using ReLU instead of MaxOut activations, and batch-normalization (as we did), the same paper reports an error of 7.40%, which is comparable to 7.43%, the error rate we report with dense fabric.

4.4.5 Discussion

In the previous section we have presented our results on three different datasets for the tasks of image classification and segmentation. In this section, keeping the results in consideration, we briefly discuss and highlight the salient points of our contribution.

One of the key advantage of the trellis is that the same architecture can be used for performing both image classification or image segmentation. The results presented in the chapter, validate this property of the trellis. In this work,

we trained the trellises separately for the two tasks. However, the trellis construction, allows for joint learning of both tasks, image classification and image segmentation.

For the task of image segmentation on Part Label dataset, we obtain results competitive to the state-of-the-art, while keeping the parameter count lower than competing methods. However, for the task of image classification on MNIST and CIFAR-10 (see Table 4.7 and Table 4.12), our dense models have about an order of magnitude more parameters for similar performance as related work. We did not, however, manually handcraft the trellis, a practice currently observed by a large part of computer vision literature on CNNs. Our goal in this work is to automate the current practice of hand-crafting architectures for specific tasks or datasets. From the results presented in this chapter, we show that the our fabrics are able to circumvent eight of the ten CNN architecture-related hyperparameters.

There are two hyperparameters for the neural fabrics, number of layers and channels. As can be seen from experimental results, performance of the trellis for both CNF-sparse and CNF-dense varies smoothly over the hyperparameter space. Hence, in practice, choice of the only two hyper-parameters of our model is not critical, as long as a large enough trellis is used.

Fabrics, compared to standard CNN architectures, are more expensive to train. This is due to the fact that by construction our fabrics are larger than any of the embedded CNNs. The training time of a trellis scales linearly with the number of nodes. The benefit is that for a linear increase in computation and memory budget, we are able to explore an exponentially large number of CNN architectures in parallel. This is an extremely advantageous trade-off as compared to one-by-one hand-crafting and testing individual architectures. To appreciate the number of embedded architectures consider the fabrics used for MNIST and CIFAR10, with 8 layers and 6 scales, the trellis embeds $7,442$ conventional CNN architectures.

Despite the fact that trellises are more expensive to train, we consider our fabric models practical. All the experiments reported in this work were conducted on a single consumer-grade Nvidia Titan X GPU.

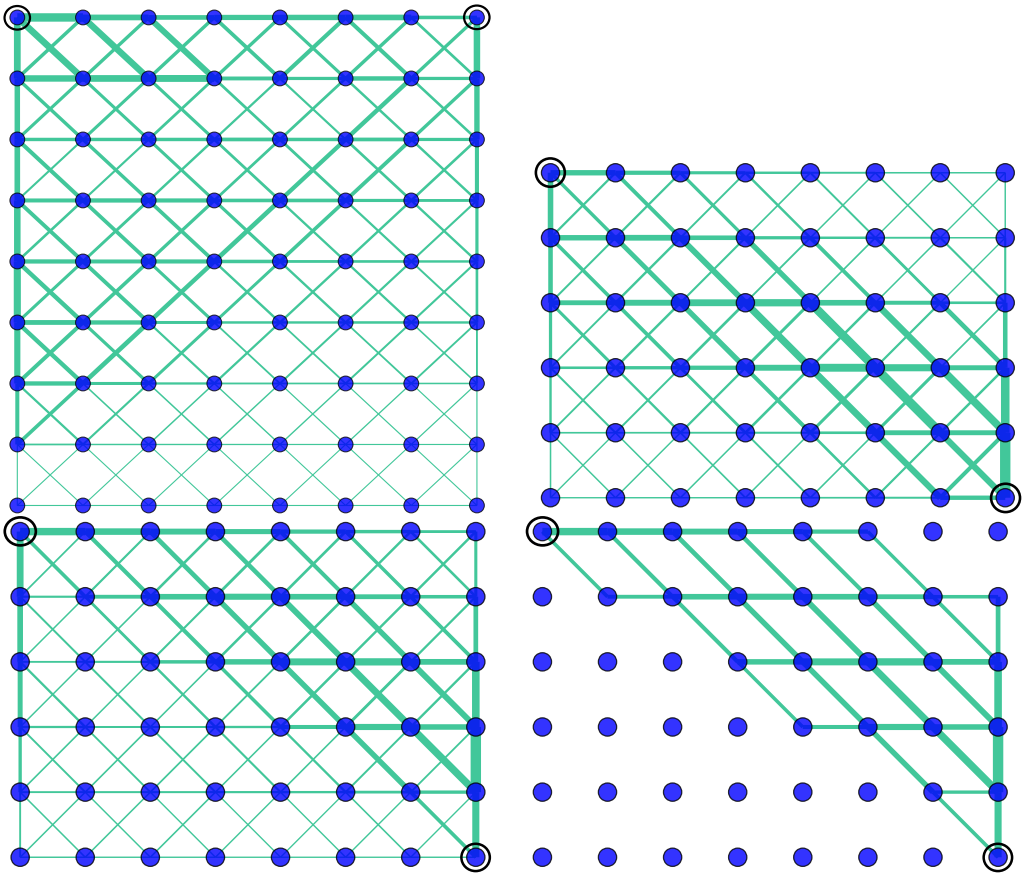


Figure 4.21 – Visualization of densely connected trellis model for Part Labels (top-left), MNIST (top-right) and CIFAR10 (bottom-left). The trellis models correspond to the best architecture found for each dataset. Layers are laid out horizontally, and scales vertically. The circled nodes represent input and output nodes of the trellis. In the bottom-right we visualize the CIFAR10 trellis after pruning. See text for details.

4.4.6 Visualization

In this section we illustrate the structure of learned trellis models. We visualize the learned trellis by plotting the edge width proportional to mean-squared filter weights (mean computed along weights and channels). In Figure 4.21 we visualize the weight strengths of learned trellis models for different datasets, and observe qualitative differences between them.

For the task of image segmentation, Part Labels model (center) immediately distributes the signal across the scale pyramid, *i.e*. across the first layer of the trellis. The information from multi-scale signal is progressively refined and aggregated towards the output node.

For the task of image classification, we have two models, CIFAR10 (left) and MNIST (right). In the case of MNIST, signal in the trellis is propagated in a band-diagonal pattern, exploiting multiple scales in each layer. With CIFAR10 we observe a similar pattern, but with a small difference. The signal is first processed at the finest scale for a few layers, and then propagated down. This is possibly due to the fact that CIFAR10 images have more variability than those in MNIST, and the first few layers extract discriminative content before downsampling and propagating the signal. These results demonstrate that even for the same task, the trellis is able to configure itself in order to accommodate nuances of the dataset.

To reduce the number of parameters of a trained trellis model, we prune the convolutions which have ℓ_2 norm of the weights below a threshold. The threshold is set as the ℓ_2 norm of convolutions averaged over the entire trellis. We evaluated this approach for CIFAR10 dataset and reduced the number of parameters from 32M to 6M in the best architecture (dense model), increasing the error from 7.43% to 9.6%. We reduced the error rate to 8.11% by fine-tuning the pruned trellis model. The number of parameters of this network is of a similar magnitude as the state of the art. We visualize the topology of pruned network in bottom-right panel of Figure 4.21, which roughly corresponds to the upper-right triangle of the fabric in the bottom-left panel of Figure 4.21, with all up-sampling connections being removed as well.

4.5 Conclusion

In this chapter we presented convolutional neural fabrics: homogeneous and sparsely connected three-dimensional trellises over response maps. The fabric

subsumes a large class of (de)convolutional networks. It sidesteps the tedious process of specifying, training, and testing individual networks in order to find good architectures. The fabric has only two main design parameters: the number of layers and the number of channels. In practice their setting is not critical: we just need a large enough fabric with enough capacity. We propose a variant with dense channel connectivity, and one with channel-doubling over scales. The latter strikes a very attractive capacity/memory trade-off.

In our experiments we study performance of image classification on MNIST and CIFAR10, and of semantic segmentation on Part Labels. We obtain excellent results that are close to the best reported results in the literature on all three datasets. These results demonstrate that our generic fabric approach is competitive with the best hand-crafted CNN architectures. We expect that these results can be further improved by using better optimization schemes such as Adam[Kingma and Ba, 2015], using dropout [Srivastava et al., 2014] or dropconect [Wan et al., 2013] regularization, and using MaxOut units [Goodfellow et al., 2013] and/or residual units [He et al., 2016] to facilitate training of deep fabrics with many channels.

Chapter 5

Conclusion

Contents

In this thesis we have focused on methods which address and benefit from the consequences of rapid increase in the amount of available digital media content. To improve the efficiency and accuracy of retrieval, we have presented a coordinated local metric learning approach in Chapter 2. In Chapter 3, we have benchmarked various strategies involving metric learning and CNNs pre-trained on visible spectrum images for heterogeneous face recognition. Finally, in Chapter 4, we have presented convolutional neural fabric to address the architecture selection problem in CNNs. We now present a summary of our contributions and results, and conclude the thesis with perspectives for future research.

5.1 Summary of Contributions

Coordinated Local Metric Learning

In Chapter 2, we have focussed on distance metric learning. Learning a task relevant metric is crucial as metrics play a crucial role in a wide range of applications in computer vision, *e.g*. local descriptor matching, fine-grained object comparison, face verification, *etc*. We presented our coordinated local metric learning (CLML) approach which learns local Mahalanobis metrics, and integrates them in a global representation where the ℓ_2 distance can be used. This allows for data visualization in a single view, and use of efficient ℓ_2-based retrieval methods. We show that our proposed approach can be interpreted as learning a linear projection on top of an explicit high-dimensional embedding

of a kernel. This interpretation allows for the use of existing frameworks for Mahalanobis metric learning for learning local metrics in a coordinated manner. We validate the effectiveness of our proposed approach through extensive experiments. Our approach improves over global metric learning and other local metric learning approaches evaluated over different features (LBP, FV-SIFT, CNN), projection dimensions, and performance measures for the task of face retrieval. Our approach also allows efficient multiple-assignment retrieval, which gives a better speed-accuracy trade-off than earlier work for a large-scale dataset with a million distractor faces.

Heterogeneous Face Recognition

Heterogeneous face recognition is the problem of recognizing faces across modalities. In most cases, the gallery of known individuals consists of normal visible spectrum images. Probe images may be forensic sketches or thermal infrared images which are useful in a forensic context or covert non-intrusive night-time acquisition respectively. In Chapter 3, we explore the use of CNNs pre-trained on visible spectrum images for the task of heterogeneous face recognition. We evaluate different metric learning strategies to reduce the discrepancies between the modalities, and find that metric learning over the features from the intermediate layers of networks is most effective. In our experiments we found that the depth of the optimal features for a given modality, is positively correlated with the domain shift between the source domain (CNN training data) and the target domain. Experimental results show the that we can use CNNs trained on visible spectrum images to obtain results that improve over the state-of-the art for heterogeneous face recognition with near-infrared images and sketches.

Convolutional Neural Fabrics

Instead of manually designing compact CNN architectures with maximum performance, our goal in Chapter 4 is to automate the current practice of handcrafting architectures for specific tasks or datasets. In our work, we propose a "fabric" that embeds an exponentially large number of CNN architectures. The fabrics sidesteps the tedious process of specifying, training and testing individual networks in order to find good architectures. The standard classification and segmentation architectures can be recovered from the fabric for different choices of hyperparamters and by setting certain weights to zero. Learning can thus efficiently configure the fabric to implement each one of exponentially many architectures and, more generally, ensembles of them. While scaling linearly in terms of computation and memory requirements, fabrics leverages exponentially

many chain-structured architectures in parallel by massively sharing weights between them. The fabric circumvents 8 out of 10 hyperparameters of the CNN architecture-related hyperparameters. Our experiments show that the two remaining hyperparameters, number of layers and channels, are not critical as long as they are large enough. In our experimental evaluation for image classification and segmentation, fabrics obtain results competitive with the state-of-the art and validate the effectiveness of our approach.

5.2 Future research prespectives

In the following three sections, we give possible extensions to the work presented in Chapters 2, 3 and 4, respectively.

Coordinated local metric learning In Chapter 2, we propose to embed local metrics in a global low dimensional representation. Our approach for learning coordinated local metrics can be interpreted as learning a linear projection on top of a expanded data representation. The expanded data representation can also be viewed as a Fisher vector image representation for GMMs [Sánchez et al., 2013], when considering just the Fisher vector components corresponding to the Gaussian means. This suggests extensions of CLML by including the FV components related to the variance parameters of the GMM, which will result in locally quadratic mappings. More generally, our approach can be applied to other generative models other than Gaussian mixture models, by performing metric learning on the FV obtained for these models. In some of our experiments we use CLML on top of FV image descriptors; In this case the expanded data representation can be interpreted as a stacked fisher vector [Simonyan et al., 2013b].

In the context of CNN, CLML has strong similarities with Maxout Networks [Goodfellow et al., 2013]. We use GMMs to obtain the soft-distribution over the local metrics, whereas Maxout units use a hard assignment by setting the output unit as the $\max$ computed over the input units. Maxout units were shown to be an piecewise linear approximation to arbitrary convex functions, and hence can learn activation functions for each hidden unit. In contrast, we show that using the locally linear projections can be used to learn a non-linear mapping by coordinating the local metrics. Our approach, CLML, can be interpreted as a soft parametric version of Maxout units. Hence, an interesting future research direction would be to explore the use of locally linear projections in a CNN as a learnable non-linear projection.

Heterogeneous face recognition In Chapter 3, we propose to leverage CNNs pre-trained on large scale visible spectrum images for the task of heterogeneous face recognition. In our work, we evaluated fine-tuning the CNN on the target domain, followed by metric learning over features from intermediate layers of the network. This approach is sub-optimal as we fine-tune the CNN for classification, followed by learning a distance metric for verification. To mitigate this issue, a direct extension of our work would be to evaluate fine-tuning the CNN directly for metric learning instead of the current two step training protocol.

Similar in spirit to Ganin and Lempitsky [2015], an interesting line of work would be to use an adversarial loss for training the CNN, but on a heterogeneous dataset. In our experiments, we evaluated fine-tuning the CNN on the heterogeneous dataset with the classification loss, but obtained limited success in fine-tuning more than one or two layers of the network. To aid in learning, an adversarial loss [Goodfellow et al., 2014] can be attached to the penultimate layer of the network which would guide the training to obtain domain-invariant representations from the network.

Convolutional neural fabrics In Chapter 4, we propose Convolutional neural fabrics which embeds an exponentially large number of CNN architectures. In our study, we have focussed on architecture learning and mainly ignored the computational and memory cost. Therefore, a desirable future work direction is to explore methods that enable us to learn architectures with a bounded computational and memory cost. In our experiments, we show that pruning edges of a trained trellis and fine-tuning the resulting architecture, reduced the parameter count by 80 % for a modest increase in error rate. Also, we have observed that the trellis recovers the optimal network architecture within the early epochs of SGD training. This hints at early pruning of trellis. Therefore, an interesting line of work would be to train the fabrics in a dynamic nature, *i.e*. by growing and pruning the trellises iteratively [Chen et al., 2016]. The key questions would be: Which edges are to be pruned? Where should the new nodes be placed? Interestingly, this procedure is analogous to synaptic pruning [Chechik et al., 1998, 1999], present in biological brains, where the synapses in the brain are pruned between the early childhood and onset of puberty.

In the current set of experiments we have used data-augmentation and weight decay for regularization during training. A promising direction is to investigate the use of more explicit forms of regularization in the fabric. In the same spirit, Larsson et al. [2016] train an ensemble of multiple networks of varying depth in parallel. The networks are linked to each other at different mid-level layers.

They propose drop-path regularization, which prevents co-adaptation of parallel paths in a network. We can use similar path based regularization schemes to encourage different sub-paths in a trellis during training.

Bibliography

R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012. 31

R. Arandjelovic and A. Zisserman. All about vlad. In *CVPR*, 2013. 76

Y. Aytar and A. Zisserman. Tabula rasa: Model transfer for object category detection. In *ICCV*, 2011. 49

J. Ba and R. Caruana. Do deep nets really need to be deep? In *NIPS*, 2014. 56

A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky. Neural codes for image retrieval. In *ECCV*, 2014. 50, 54, 55, 79

P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pages 711–720, 1997. 15

A. Bellet, A. Habrard, and M. Sebban. A Survey on Metric Learning for Feature Vectors and Structured Data. *ArXiv e-prints*, 1306.6709, 2013. 15

S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *NIPS*, 2007. 52

H. S. Bhatt, S. Bharadwaj, R. Singh, and M. Vatsa. Memetically optimized MCWLD for matching sketches with digital face images. *Transactions on Information Forensics and Security*, 7(5):1522–1535, 2012. 73

B. Bhattarai, G. Sharma, F. Jurie, and P. Pérez. Some faces are more equal than others: Hierarchical organization for accurate and efficient large-scale identity-based face retrieval. In *ECCV Workshops*, 2014. vii, 11, 12, 13, 21, 23, 24, 31, 32, 35, 36, 37, 38

J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2006. 45

J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman. Learning bounds for domain adaptation. In *NIPS*, 2008. 45

J. Blitzer, S. Kakade, and D. P. Foster. Domain adaptation with coupled subspaces. In *AISTATS*, 2011. 45

J. Bohné, Y. Ying, S. Gentric, and M. Pontil. Large margin local metric learning. In *ECCV*, 2014. 11, 12, 21, 23, 24, 35

J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah. Signature verification using a siamese time delay neural network. In *NIPS*, 1993. 11

M. Brown, G. Hua, and S. Winder. Discriminative learning of local image descriptors. *PAMI*, 33(1), 2011. 76

Q. Cao, Y. Ying, and P. Li. Similarity metric learning for face recognition. In *ICCV*, 2013. 12, 40

J.-R. Chang and Y.-S. Chen. Batch-normalized maxout network in network. Arxiv preprint, 2015. 106, 108

G. Chechik, I. Meilijson, and E. Ruppin. Synaptic pruning in development: a computational account. *Neural computation*, 1998. 116

G. Chechik, I. Meilijson, and E. Ruppin. Neuronal regulation: A mechanism for synaptic pruning during brain maturation. *Neural Computation*, 1999. 116

G. Checkik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *JMLR*, 11:1109–1135, 2010. 18

D. Chen, X. Cao, F. Wen, and J. Sun. Blessing of dimensionality: high dimensional feature and its efficient compression for face verification. In *CVPR*, 2013. 38, 39

L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *ICLR*, 2015. 88, 91

T. Chen, I. Goodfellow, and J. Shlens. Net2net: Accelerating learning via knowledge transfer. In *ICLR*, 2016. 78, 79, 116

X. Chen, P. Flynn, and K. Bowyer. IR and visible light face recognition. *CVIU*, 99(3):332–358, 2005. 46

J. Choi, S. Hu, S. S. Young, and L. S. Davis. Thermal to visible face recognition. In *SPIE Defense, Security, and Sensing*, 2012. 65

S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005. 11, 12

E. J. Crowley, O. M. Parkhi, and A. Zisserman. Face painting: querying art with photos. In *British Machine Vision Conference*, 2015. 58

G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *ECCV Int. Workshop on Stat. Learning in Computer Vision*, 2004. 3

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. doi: 10.1109/CVPR.2005.177. URL `http://hal.inria.fr/inria-00548512`. 58, 76

H. Daumé III. Frustratingly easy domain adaptation. *ACL 2007*, 2007. 45

H. Daume III and D. Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26, 2006. 45

J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. Information-theoretic metric learning. In *ICML*, 2007. 11, 15, 16, 52

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 3

T. I. Dhamecha, P. Sharma, R. Singh, and M. Vatsa. On effectiveness of histogram of oriented gradient features for visible to near infrared face matching. In *ICPR*, 2014. 71

J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014. 79

A. Dosovitskiy, J. Springenberg, M. Riedmiller, and T. Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *NIPS*, 2014. 9

L. Duan, D. Xu, and I. W. Tsang. Learning with augmented features for heterogeneous domain adaptation. In *ICML*, 2012. 49

M. Everingham, J. Sivic, and A. Zisserman. 'Hello! My name is... Buffy' - automatic naming of characters in TV video. In *BMVC*, 2006. 65

M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop, 2007. 54

M. Everingham, J. Sivic, and A. Zisserman. Taking the bite out of automatic naming of characters in TV video. *Image and Vision Computing*, 27(5):545–559, 2009. 31

M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, 2012. 54

C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *PAMI*, 35(8):1915–1929, 2013. 89, 91, 98

B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Subspace alignment for domain adaptation. In *ICCV*, 2013. xiii, 45, 48, 49, 51, 52, 70, 71

B. Fernando, A. Habrard, M. Sebban, and T. Tuytelaars. Subspace alignment for domain adaptation. *arXiv preprint arXiv:1409.5241*, 2014. 52

P. Fischer, A. Dosovitskiy, and T. Brox. Descriptor matching with convolutional neural networks: a comparison to sift. *CoRR, arXiv:1405.5769*, 2014. 88

R. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936. 15

A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*, 2007. 11, 21

Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015. viii, 50, 55, 116

Z. Ghahramani and G. Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, University of Toronto, May 1996. 28

R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 88

A. Globerson and S. Roweis. Metric learning by collapsing classes. In *NIPS*, 2006. 11, 15

B. Gong, Y. Shi, F. Sha, and K. Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, 2012. 48, 49

I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *ICML*, 2013. 97, 102, 106, 107, 108, 112, 115

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014. 116

R. Gopalan, R. Li, and R. Chellappa. Domain adaptation for object recognition: An unsupervised approach. In *ICCV*, 2011. 48, 49

A. Graves, S. Fernández, and J. Schmidhuber. Multi-dimensional recurrent neural networks. In *Proceedings of the International Conference on Artificial NeuralNetworks*, 2007. 94

M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? Metric learning approaches for face identification. In *ICCV*, 2009. 11, 12, 13, 15, 16, 29, 39, 46, 62, 63

M. Guillaumin, J. Verbeek, and C. Schmid. Multiple instance metric learning from automatically labeled bags of faces. In *ECCV*, 2010. 11, 12, 16

S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *ECCV*, 2014. 89

H. Han, B. F. Klare, K. Bonnen, and A. K. Jain. Matching composite sketches to face photos: A component-based approach. *Transactions on Information Forensics and Security*, 8(1):191–204, 2013. 65

D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12), 2004. 52, 60

B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *ECCV*. Springer, 2014. 89

B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 89, 93

S. Hauberg, O. Freifeld, and M. Black. A geometric take on metric learning. In *NIPS*, 2012. 21, 24, 28

K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014. 88

K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. Arxiv preprint, 2016. 108, 112

J. J. Heckman. Sample selection bias as a specification error. *Econometrica: Journal of the Econometric Society*, 1979. 45

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2014. 56

S. Honari, J. Yosinski, P. Vincent, and C. Pal. Recombinator networks: Learning coarse-to-fine feature aggregation. In *CVPR*, 2016. xi, 93

S. Hong, H. Noh, , and B. Han. Decoupled deep neural network for semi-supervised semantic segmentation. In *NIPS*, 2015. 88

Y. Hong, Q. Li, J. Jiang, and Z. Tu. Learning a mixture of sparse distance metrics for classification and dimensionality reduction. In *ICCV*, 2011. 11, 21

H. Hotelling. Relation betweeen two sets of variates. *Biometrika*, 28:322–377, 1936. 51, 52

G. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: a database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, 2007. viii, ix, 12, 30, 44, 45, 64, 101

J. Huang, S. R. Kumar, M. Mitra, W.-J. Zhu, and R. Zabih. Spatial color indexing and applications. *IJCV*, 35(3), 1999. 3

Y. Huang, C. Li, M. Georgiopoulos, and G. Anagnostopoulos. Reduced-rank local distance metric learning. In *ECML*, 2013. 11, 21, 22, 24, 35

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 102

T. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *NIPS*, 1999. 28

N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent data analysis*, 6(5), 2002. 45

H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010. 3

H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *PAMI*, 33(1):117–128, 2011. 12, 24, 29, 76

J. Jiang and C. Zhai. Instance weighting for domain adaptation in nlp. In *Association for Computational Linguistics*, 2007. 45

Y. Jin, J. Lu, and Q. Ruan. Large margin coupled feature learning for cross-modal face recognition. In *International Conference on Biometrics*, 2015. 59, 71

F. Juefei-Xu, D. Pal, and M. Savvides. NIR-VIS heterogeneous face recognition via cross-spectral joint dictionary learning and reconstruction. In *Computer Vision and Pattern Recognition Workshops*, 2015. 51, 57, 71

A. Kae, K. Sohn, H. Lee, and E. Learned-Miller. Augmenting CRFs with Boltzmann machine shape priors for image labeling. In *CVPR*, 2013a. 101, 103, 104

A. Kae, K. Sohn, H. Lee, and E. Learned-Miller. Augmenting crfs with boltzmann machine shape priors for image labeling. In *CVPR*, 2013b. 78

N. Kalchbrenner, I. Danihelka, and A. Graves. Grid long short-term memory. In *ICLR*, 2016. 94, 105, 106

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 112

B. Klare and A. Jain. Heterogeneous face recognition: Matching NIR to visible light images. In *ICPR*, 2010. 58

B. Klare, Z. Li, and A. Jain. Matching forensic sketches to mug shot photos. *PAMI*, 33(3):639–646, 2011. 45

S. Kong, J. Heo, B. Abidi, J. Paik, and M. Abidi. Recent advances in visual and infrared face recognition – a review. *CVIU*, 97(1):103 – 135, 2005. ISSN 1077-3142. 45

M. Köstinger, M. Hirzer, P. Wohlhart, P. Roth, and H. Bischof. Large scale metric learning from equivalence constraints. In *CVPR*, 2012. 10, 11, 12, 15, 17, 62

A. Krizhevsky. Learning multiple layers of features from tiny images. 2009. 78, 102

A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. x, 3, 53, 54, 55, 76, 79, 84, 85, 86, 88, 89, 92, 98

B. Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2012. 15

P. Kulkarni, J. Zepeda, F. Jurie, P. Pérez, and L. Chevallier. Learning the structure of deep architectures using l1 regularization. In *BMVC*, 2015. 93

P. L. Lai and C. Fyfe. Kernel and nonlinear canonical correlation analysis. *International Journal of Neural Systems*, 10(05):365–377, 2000. 52

R. Lajugie, F. Bach, and S. Arlot. Large-margin metric learning for constrained partitioning problems. In *ICML*, 2014. 10

G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016. 116

Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS*, 1989. 3, 76, 79

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, pages 2278–2324, 1998. 78, 79, 102

C.-Y. Lee, P. Gallagher, and Z. Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *AISTATS*, 2015. 106, 108

C.-Y. Lee, P. Gallagher, and Z. Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *AISTATS*, 2016. 93

C. J. Leggetter and P. C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer Speech & Language*, 1995. 45

Z. Lei and S. Li. Coupled spectral regression for matching heterogeneous faces. In *CVPR*, 2009. 52

F. Li, S. J. Pan, O. Jin, Q. Yang, and X. Zhu. Cross-domain co-extraction of sentiment and topic lexicons. In *Association for Computational Linguistics*, 2012. 45

S. Li, D. Yi, Z. Lei, and S. Liao. The CASIA NIR-VIS 2.0 face database. In *Computer Vision and Pattern Recognition Workshops*, 2013. 46, 60, 64, 71

S. Liao, Z. Lei, D. Yi, and S. Li. A benchmark study of large-scale unconstrained face recognition. In *International Joint Conference on Biometrics*, 2014. 12, 31

G. Lin, C. Shen, A. van den Hengel, and I. Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *CVPR*, 2016. 88, 91

M. Lin, Q. Chen, and S. Yan. Network in network. Arxiv preprint, 2013. 102, 106, 108

M. Lin, Q. Chen, and S. Yan. Network in network. In *ICLR*, 2014a. 87

T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755, 2014b. 54

S. Liu, J. Yang, C. Huang, , and M.-H. Yang. Multi-objective convolutional learning for face labeling. In *CVPR*, 2015. 104

J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. x, 79, 83, 88, 89, 90, 91, 93, 98

J. L. Long, N. Zhang, and T. Darrell. Do convnets learn correspondence? In *NIPS*, 2014. 88

D. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999. 76

J. Lu, V. Liong, X. Zhou, and J. Zhou. Learning compact binary face descriptor for face recognition. *PAMI*, 2015. 71

M. J. Lyons, S. Akamatsu, M. Kamachi, J. Gyoba, and J. Budynek. The japanese female facial expression (jaffe) database, 1998. viii, 44

J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional kernel networks. In *NIPS*, 2014. 106

A. Martinez and R. Benavente. The AR face database. Technical report, 1998. 65

T. Melzer, M. Reiter, and H. Bischof. Nonlinear feature extraction using generalized canonical correlation analysis. In *ICANN*, pages 353–360, 2001. 52

A. Mignon and F. Jurie. CMML: a new metric learning approach for cross modal matching. In *ACCV*, 2012a. 45, 51, 53, 59, 63

A. Mignon and F. Jurie. PCCA: A new approach for distance learning from sparse pairwise constraints. In *CVPR*, 2012b. 11, 12, 15, 35, 53

L. Mihalkova, T. Huynh, and R. J. Mooney. Mapping and revising markov logic networks for transfer learning. In *AAAI*, 2007. 50

F. Milletari, N. Navab, and S.-A. Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. *CoRR, arXiv:1606.04797*, 2016. 91

D. Mishkin and J. Matas. All you need is a good init. In *ICLR*, 2016. 107, 108

I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stich networks for multi-task learning. In *CVPR*, 2016. 92

P. Mittal, A. Jain, G. Goswami, R. Singh, and M. Vatsa. Recognizing composite sketches with digital face images via ssd dictionary. In *International Joint Conference on Biometrics*, 2014. 46, 65, 73

P. Mittal, M. Vatsa, and R. Singh. Composite sketch recognition via deep network-a transfer learning approach. In *International Conference on Biometrics*, 2015. 72, 73, 74

G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *NIPS*, pages 2924–2932, 2014. 20

V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 84, 85

F. Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, and P. E. Barbano. Toward automatic phenotyping of developing embryos from videos. *IEEE Transactions on Image Processing*, 2005. 89

H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015. xi, 76, 88, 90, 91

Y.-K. Noh, B.-T. Zhang, and D. Lee. Generative local metric learning for nearest neighbor classification. In *NIPS*, 2010. 11, 21, 22

E. Nowak and F. Jurie. Learning visual similarity measures for comparing never seen objects. In *CVPR*, 2007. 10, 12

T. Ojala, M. Pietikäinen, and T. Mäenpää. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *PAMI*, 24(7): 971–987, 2002. 12, 31, 58, 76

M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014. 45, 48, 50, 54, 55

S. Pan and Q. Yang. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.*, 22(10):1345–1359, 2010. doi: 10.1109/TKDE.2009.191. URL http://dx.doi.org/10.1109/TKDE.2009.191. 49, 50

O. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *BMVC*, 2015. 39, 40, 45, 54, 59, 62

V. Patel, R. Gopalan, R. Li, and R. Chellappa. Visual domain adaptation: a survey of recent advances. *IEEE Signal Processing Magazine*, 32(3):53 – 69, 2015. 49

K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, pages 559–572, 1901. 14

A. Pentina and C. H. Lampert. A pac-bayesian bound for lifelong learning. In *ICML*, 2014. 50

F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007. 3

F. Perronnin, J. Sánchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010. 76

T. Pfister, J. Charles, and A. Zisserman. Flowing ConvNets for human pose estimation in videos. In *CVPR*, 2015. 88

J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007. 76

J. Philbin, M. Isard, J. Sivic, and A. Zisserman. Descriptor learning for efficient retrieval. In *ECCV*, 2010. 76

P. H. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene labeling. In *ICML*, 2014. 89

B. Riggan, C. Reale, and N. Nasrabadi. Coupled auto-associative neural networks for heterogeneous face recognition. *IEEE Access*, 3:1620–1632, 2015. 59, 71

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, 2015. xi, 76, 91, 98

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 2000. 28

Y. Rui, T. S. Huang, and S.-F. Chang. Image retrieval: Current techniques, promising directions, and open issues. *Journal of visual communication and image representation*, 10(1), 1999. 3

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 115(3), 2015. 53

K. Saenko, B. Kulis, M. Fritz, and T. Darrell. Adapting visual category models to new domains. In *ECCV*, 2010. viii, 47, 48, 52

J. Sánchez, F. Perronnin, and T. de Campos. Modeling the spatial layout of images beyond spatial pyramids. *Pattern Recognition Letters*, 33(16):2216–2223, 2012. 31

J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the Fisher vector: Theory and practice. *IJCV*, 105(3):222–245, 2013. 28, 31, 115

M. Sarfraz and R. Stiefelhagen. Deep perceptual mapping for thermal to visible face recognition. In *BMVC*, 2015. 57, 65

S. Saxena and J. Verbeek. Coordinated local metric learning. In *ICCV ChaLearn Looking at People Workshops*, 2015. 6

S. Saxena and J. Verbeek. Convolutional Neural Fabrics. In *NIPS*, 2016a. 8

S. Saxena and J. Verbeek. Heterogeneous Face Recognition with CNNs. In *ECCV TASK-CV Workshops*, 2016b. 7

B. Schölkopf, A. J. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998. 20

F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, 2015. 39, 40, 45, 54

P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014. 54, 86, 88

A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014. 3, 48, 50, 54, 79

S. Shekhar, V. M. Patel, H. V. Nguyen, and R. Chellappa. Generalized domain-adaptive dictionaries. In *CVPR*, 2013. 49

Y. Shi, A. Bellet, and F. Sha. Sparse compositional metric learning. In *AAAI*, 2014. 11, 13, 21, 22, 24, 35, 37

H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2), 2000. 45

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 40

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. xi, 60, 79, 85, 87, 88, 89, 90, 97, 101

K. Simonyan, O. Parkhi, A. Vedaldi, and A. Zisserman. Fisher vector faces in the wild. In *BMVC*, 2013a. 12, 31, 38, 39, 45, 59, 62, 76

K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Fisher networks for large-scale image classification. In *NIPS*, 2013b. 115

S. Singh, D. Hoiem, and D. Forsyth. Swapout: learning an ensemble of deep architectures. In *NIPS*, 2016. 94

N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM transactions on graphics (TOG)*, 2006. 75

J. Springenberg, A. D. andT. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *ICLR*, 2015. 93, 107, 108

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 2014. 94, 102, 112

Y. Sun, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *CVPR*, 2014. 12

M. J. Swain and D. H. Ballard. Color indexing. *IJCV*, 7(1), 1991. 3

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015a. x, 86, 87, 88, 89

C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. *CoRR, arXiv:1512.00567*, 2015b. 81, 87

Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. DeepFace: Closing the gap to human-level performance in face verification. In *CVPR*, 2014. 12, 39, 40, 45

X. Tan and B. Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. In *International Workshop on Analysis and Modeling of Faces and Gestures*, 2007. 60

Y. Teh and S. Roweis. Automatic alignment of local representations. In *NIPS*, 2003. 28

S. Thrun and L. Pratt. *Learning to learn*. Springer Science & Business Media, 2012. 50

T. Tommasi, F. Orabona, and B. Caputo. Safety in numbers: Learning categories from few examples with multi model knowledge transfer. In *CVPR*, 2010. 49

A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR*. IEEE, 2011. 44

L. Torrey and J. Shavlik. *Transfer learning, in: Soria, E., Martin, J.,Magdalena, R., Martinez, M., Serrano, A. (Eds.), Handbook of Research on Machine Learning Applications*. IGI Global, 2009. viii, 50

S. Trivedi, D. Mcallester, and G. Shakhnarovich. Discriminative metric learning by neighborhood gerrymandering. In *NIPS*, 2014. 11, 18

S. Tsogkas, I. Kokkinos, G. Papandreou, and A. Vedaldi. Deep learning for semantic part segmentation with high-level guidance. Arxiv preprint, 2015. 88, 104

E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014. 55

E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko. Simultaneous deep transfer across domains and tasks. In *ICCV*, 2015. ix, 50, 55, 56, 57

T. Van Gestel, J. A. Suykens, J. De Brabanter, B. De Moor, and J. Vandewalle. Kernel canonical correlation analysis and least squares support vector machines. In *ICANN*, pages 384–389, 2001. 52

L. Wan, M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus. Regularization of neural networks using DropConnect. In *ICML*, 2013. 105, 106, 108, 112

C. Wang and S. Mahadevan. Heterogeneous domain adaptation using manifold alignment. In *IJCAI*, 2011. 49

J. Wang, A. Kalousis, and A. Woznica. Parametric local metric learning for nearest neighbor classification. In *NIPS*, 2012. 11, 21, 23, 24

J. Wang, K. Sun, F. Sha, S. Marchand-Maillet, and A. Kalousis. Two-stage metric learning. In *ICML*, 2014. 15

X. Wang and X. Tang. Face photo-sketch synthesis and recognition. *PAMI*, 2009. 51, 57, 60

K. Weinberger and L. Saul. Distance metric learning for large margin nearest neighbor classification. *JMLR*, 10:207–244, 2009. vii, 11, 15, 18, 21, 24, 58

K. Weinberger, J. Blitzer, and L. Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, 2006. 17, 21

M. Xiao and Y. Guo. Feature space independent semi-supervised domain adaptation via kernel matching. *PAMI*, 2015. 49

M. Yamada, L. Sigal, and M. Raptis. No bias left behind: Covariate shift adaptation for discriminative 3d pose estimation. In *ECCV*, 2012. 45

Y.-R. Yeh, C.-H. Huang, and Y.-C. F. Wang. Heterogeneous domain adaptation and classification by exploiting the correlation subspace. *IEEE Transactions on Image Processing*, 23(5):2009–2018, 2014. viii, 51

D. Yi, R. Liu, R. Chu, Z. Lei, and S. Z. Li. Face matching between near infrared and visible light images. In *International Conference on Biometrics*, 2007. 52

D. Yi, Z. Lei, S. Liao, and S. Li. Learning face representation from scratch. In *Arxiv preprint*, 2014. 12, 31, 32, 39, 40, 45, 46, 60, 62, 67

D. Yi, Z. Lei, and S. Z. Li. Shared representation learning for heterogeneous face recognition. In *International Conference on Automatic Face and Gesture Recognition*, 2015. 71

B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *ICML*, 2004. 45

M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 86, 90

M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, 2011. 90

D.-C. Zhan, M. Li, Y.-F. Li, and Z.-H. Zhou. Learning instance specific distances using metric propagation. In *ICML*, 2009a. 11, 21

D.-C. Zhan, M. Li, Y.-F. Li, and Z.-H. Zhou. Learning instance specific distances using metric propagation. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009b. 21

H. Zheng, Y. Liu, M. Ji, F. Wu, and L. Fang. Learning high-level prior with convolutional neural networks for semantic segmentation. Arxiv preprint, 2015. 88, 104

B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014. 44, 54

Y. Zhou, X. Hu, and B. Zhang. Interlinked convolutional neural networks for face parsing. In *International Symposium on Neural Networks*, 2015. 92