UNIVERSITY OF CALIFORNIA,
IRVINE

Learning Robust Features and Metrics for Image Classification and Matching

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Bailey Kong

Dissertation Committee:
Professor Charless C. Fowlkes, Chair
Professor Padhraic Smyth
Professor Xiaohui Xie

2018

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

First and foremost, I want to thank my advisor, Prof. Charless Fowlkes. He first took me under his wing as an undergraduate when I knew nothing about research and little about computer vision. His enthusiasm for research—curiosity for understanding how things work and explaining concepts simply—is contagious. It was his enthusiasm that led me to decide I wanted to pursue a Ph.D, where I had the fortunate opportunity to continue learning from him as a doctoral student. There is no doubt in my mind that without his continued guidance, patience, and support, I would not be the student and scholar I am today, and for that, I am sincerely grateful.

I especially want to thank James and Phuc for the endless conversations on the meaning of life, research, and everything in between. And my friends: Daeyun, Dennis, Geng, Minhaeng, Mohsen, Raúl, Sam, Shu, Xiangxin, Yi, and Zhe for their helpful discussions and company.

I want to thank my family. My mom and dad for their love and support, and for all the sacrifices they made to give me the best opportunities they could. And my brother for being a trailblazer—making my life just a little easier through watching his actions.

Finally, I want to thank to Sarena Wiesner and Yaron Shor for providing access to the Israeli shoeprint dataset, Adam Kortylewski for providing his results for comparison, and my funding sources: NSF Grants DBI-1262547 and DBI-1053036, and by Center for Statistics and Applications in Forensic Evidence (CSAFE) through NIST Cooperative Agreement #70NANB15H176.

CURRICULUM VITAE

Bailey Kong

EDUCATION

Doctor of Philosophy in Computer Science **2018**
University of California, Irvine *Irvine, California*

Bachelor of Science in Computer Science **2010**
University of California, Irvine *Irvine, California*

RESEARCH EXPERIENCE

Graduate Research Assistant **2012–2018**
University of California, Irvine *Irvine, California*

TEACHING EXPERIENCE

AppJam+ Mentor **2016–2017**
Dreams for Schools *Orange County, California*

Teaching Assistant **2013–2014**
University of California, Irvine *Irvine, California*

REFEREED CONFERENCE PUBLICATIONS

Cross-Domain Forensic Shoeprint Matching **Sep 2017**
British Machine Vision Conference

SOFTWARE

MCNCC `https://github.com/bkong/MCNCC`
An implementation of our cross-domain image matching pipeline.

FCSC `https://github.com/bkong/FCSC`
An implementation of Bristow et al.'s CVPR 2013 paper "Fast Convolutional Sparse Coding."

FSC `https://github.com/bkong/FSC`
An implementation of patch-based sparse coding and dictionary learning using ADMM.

ABSTRACT OF THE DISSERTATION

Learning Robust Features and Metrics for Image Classification and Matching

By

Bailey Kong

Doctor of Philosophy in Computer Science

University of California, Irvine, 2018

Professor Charless C. Fowlkes, Chair

Vision systems use a pipeline of feature extraction and analysis to predict the desired output from input image data. With robust features, we can achieve high accuracy using carefully chosen, but simple analysis. Learning features and using features effectively are two problems we focus on in this work. We propose a novel formulation of convolutional sparse coding called *spherical sparse coding* (SSC). SSC removes the need for iterative optimization to compute the sparse codes (features) for the typical least squares formulation. Using the SSC formulation, we show a clear connection between convolutional sparse coding and convolutional neural networks (CNNs). We extend SSC to a supervised method for classification that uses codes that are biased by a hypothesized class. These class-specific codes can be reconstructed to give us images that maximize the classification score of the hypothesized class. Next, we propose using the Siamense network architecture with the *multi-channel normalized cross-correlation* (MCNCC) similarity metric for cross-domain image matching. We show that our choices in how we use features can have significant performance consequences; even prior to learning any parameters, we achieve state-of-the-art performance using off-the-shelf CNN features with MCNCC on a number of different cross-domain matching problems.

Chapter 1

Introduction

Building highly accurate vision systems requires discovering representations from the input data and effectively using the captured information to predict the target output. Having an informative representation that is highly discriminative can allow us to use a simple predictor to get great performance, having an uninformative representation may require us to use a more sophisticated predictor to get the same performance. Consider the task of classifying 2D geometric shape in a binary image, we could try simply compiling a set of image templates (one of each shape we care about) and classify the shape by adopting the same shape label as the closest matching image in our set using a naïve metric like Euclidean distance or Cosine similarity. This simple method works well if the set of templates was representative of what we expect to see (*i.e.*, a single scale and orientation for each shape), but quickly fails when these assumptions change. So this hypothetical example method is not very robust as it does not account for many of the intraclass variances we typically see in images, *e.g.*, geometric transformations, lighting effects, and measurement noise. A more robust method could use a better representation of the data, that for example builds a histogram of corners that are greater than, equal to, and less than 90 degrees, while continuing to use a naïve distance/similarity metric. Alternatively, a different but equally robust method could use

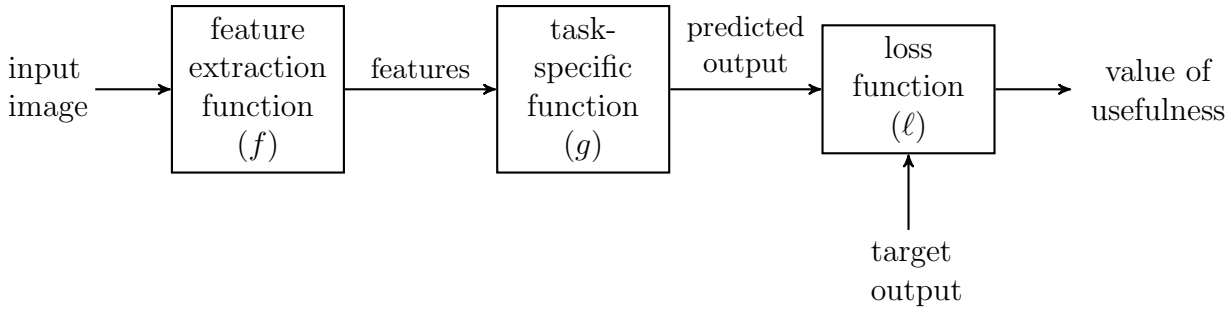


Figure 1.1: Learning pipeline.

the same set of images (one of each shape), but use a more sophisticated metric that is invariant to changes in the scale and orientation of the shape.

In the past, features and metrics were hand-designed in a laborious process of hypothesizing and evaluating their usefulness. This process, illustrated as a pipeline in Fig. 1.1, could take months or years, if ever, to meet the desired performance. Automating this process is the essence of machine learning and computer vision.

This pipeline contains three components: the feature extraction function f, the task-specific function g, and the loss function ℓ. We also have two types of data, the input image data and the target data. The target data is task specific, for example, for classification this will be a single class label for each input data or for segmentation this will be pixel-level label map for each input data. The feature extraction function takes the input image and outputs features that preserve useful properties while removing harmful properties from the input. While the properties that are useful versus harmful are task specific, features may remove variances like affine transformations, color, and lighting effects, and may select discriminating variances like shapes. The task-specific function then takes the features as input and outputs the predicted target data. The loss function then compares the predicted target data to the true target data ("ground truth") and outputs a measure of difference between the two.

Note that while we describe a pipeline where the feature extraction function and the task-specific function are separate components (piece-wise), we could describe a pipeline where

the two are merged into a single component that produces an output directly from an input (end-to-end). An end-to-end pipeline can yield higher performance over a piece-wise pipeline, but only when there is enough task-specific training data. This can be difficult to obtain for many tasks, so at the expense of some performance the piece-wise pipeline offers the flexibility of building each component separately which requires much less task-specific training data and can even exploit training data from other tasks. Consider, for example, an esoteric task like forensic shoeprint matching which has labeled data on the order of hundreds of examples, and contrast this with common tasks like object classification which has labeled data on the order of millions. Using the piece-wise pipeline we can make use of the vast amount of labeled data from object classification to learn a feature extraction function for object classification, but repurpose it for shoeprint matching.

Learning done with respect to the pipeline involves updating the parameters of the functions. And depending on our particular task, the feature extraction, task-specific, or both functions are parameterized so that we can learn the parameters that minimize the loss function. In this thesis, we will cover ways to learn parameters of both the feature extraction and task-specific functions from image data for tasks such as object classification, matching, and alignment.

1.1 Learning Features

The first part of the thesis focuses on feature learning methods. In Chapter 2 we review sparse coding, a feature learning method that finds a sparse representation that accurately reconstructs the input data as measured using least squares (ℓ_2-norm). These sparse representations (or sparse codes) are then used as features for other tasks like image classification. Sparse coding is a popular method because it is "unsupervised;" this means that no additional annotated/label data is needed to learn the features beyond the input data. The

typical formulation of sparse coding reconstructs image patches rather than whole images—which means it assumes image patches are unrelated, even when patches may be overlapping in the original image. This is remedied with the convolutional formulation which removes the assumption using the convolution operator. Both of these formulations, however, require iterative optimization due to the combination of the quadratic least squares and the linear ℓ_1-regularizer used in their formulations.

We address this limitation in Chapter 3 with *spherical sparse coding* [32], our novel formulation to the convolutional sparse coding problem. Spherical sparse coding is inspired from the observation that if the codes are unit length, then expanding least squares results in only an inner product—a linear term—that compares the input data and its reconstruction. By replacing least squares with inner product, we are able to compute the sparse codes (features) in a feed-forward way; and this new formulation establishes a clear connection between sparse coding and another popular feature learning method, convolutional neural networks.

We extend spherical sparse coding to a supervised method for classification called *energy-based spherical sparse coding* (EB-SSC). This method is inspired by biased competition theory which suggests that the visual processing system can be influenced by other top-down processes; so energy-based spherical sparse coding produces codes that are specific to a hypothesized class (class-specific features). A unique aspect of class-specific features is that we can visualize the information captured by the features. By reconstructing the input image using these features, we can actually see a class-specific version of the image that maximizes the classification score for a particular class. This work is described in detail in Chapter 3.

1.2 Learning to Use Features

While the first part of this thesis focused on learning features specific to our data and task, the second part of this thesis focuses on methods and metrics that help us utilize the information already captured by existing features. We demonstrate this in Chapter 4, where we introduce a similarity metric for cross-domain image matching called *multi-channel normalized cross-correlation* (MCNCC). MCNCC is a multi-channel extension to normalized cross-correlation (also known as Pearson's correlation coefficient) that is commonly used for template matching of gray-scale images. It was originally applied to multi-spectral images by Fisher and Oliver for single-domain image matching [14], but we show that it is useful for cross-domain image matching and CNN feature maps. In doing so, we show that MCNCC performs substantially better than commonly used Cosine similarity or other standardization techniques (*e.g.*, whitening) on features extracted from pretrained convolutional neural network models. This is demonstrated on a variety of tasks: matching forensic shoeprints to laboratory test impressions, facade images to segmentation labels, and aerial photos to map images.

Despite state-of-the-art performance, one undesirable aspect of our setup is the need to search over alignments. For example, while laboratory test impressions are always the entire shoeprint, forenic shoeprints are frequently partial prints, and so finding the best possible alignment of two images is necessary. Additionally, our pipeline evaluates each forensic shoeprint against every laboratory test impression, meaning it checks each impression to find the most similar one in the set. This poses a problem as matching is not scalable to even modest sized problems where there are thousands or more images in the database to match against.

The first step to solving these problems is aligning the database of images. There are two reasons this. Firstly, if the database of images were all aligned in a canonical pose,

then the best alignment of a forensic shoeprint to one particular laboratory test impression would be the same for all test impressions—allowing us to also replace the current alignment search function with a fast alignment predictor. Secondly, an aligned database gives us an efficient means to build a search tree for fast (sublinear) matching using existing hierarchical clustering methods. Aligning the database, or jointly estimating warping parameters for a set of images to make them as similar as possible, is often referred to as "congealing." In Chapter 5, we propose an algorithm for congealing a database of images together—coarsely by shape and then finely by shape and raw pixels. We then show that an alignment predictor can replace explicit alignment search for fast image matching on the congealed database.

Chapter 2

Sparse Coding

In the previous chapter, we discussed our motivation for learning features from images and a pipeline for doing so. We focus in this chapter on a feature learning method that aims to learn the underlying features that make up the image data itself. For such methods where the input and target data are the same, we refer to this class of methods as *unsupervised* because no additional annotation or *supervision* is necessary to use them.

2.1 Introduction

Sparse coding (SC) is the problem of reconstructing the input data using a sparse linear combination of basis vectors. Its methods are motivated by inspirations from neural coding, where encoded data is represented by the strong activation of few neurons. A seminal paper by Olshausen and Field [47] showed that sparse coding of natural images produce Gabor-like wavelet filters that resemble the receptive fields found in the primary visual cortex (V1). SC methods have been used in many different areas like: image denoising [13, 70], where it is believed that reconstructing the input using a sparse representation will produce the

original clean input as the residual of the observed signal is noise; image compression [11], where the sparse representation requires fewer bits to store or transmit than the original signal; and most relevant to this work is image classification [27, 66, 68, 69], where the sparse representation is used as features to classify objects in images.

The canonical sparse coding is formulated as a least squares reconstruction problem with a regularization term that penalizes non-sparsity of the codes using the ℓ_0-norm. Solving this formulation is NP-hard, however, and so a convex relaxation of the problem can be formulated using the ℓ_1-norm. And while the ℓ_1-norm does not explicitly measure non-sparsity, it has been shown to induce sparsity due to its shape [12, 15]. This convex relaxation of sparse coding problem is often formulated as:

$$\begin{aligned} \arg\min_{\mathbf{d},\mathbf{z}} & \frac{1}{2}\sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{D}\mathbf{z}_i\|_2^2 + \beta\|\mathbf{z}_i\|_1 \\ \text{s.t. } & \|\mathbf{d}_k\|_2 \leq 1 \qquad \forall k = 1, \ldots, K, \end{aligned} \tag{2.1}$$

where $\mathbf{x}_i \in \mathbb{R}^m$ is an input vector (*e.g.*, an image patch), $\mathbf{D} \in \mathbb{R}^{m \times n}$ is a matrix containing a set of basis vectors (typically $m < n$), $\mathbf{z}_i \in \mathbb{R}^m$ is a vector that identifies a sparse linear combination of the bases to reconstruct $\mathbf{x}_i$, and β controls the non-sparsity penalization. There are a few things to note about this formulation; the first is that this system is underdetermined as it only one equation with $K + 1$ unknowns, which is why unit length constraints on the basis vectors is required to find an unique solution. The second is that while the problem of solving for each $\mathbf{z}_i$ when $\mathbf{d}$ is known and visa-versa is convex, the problem of jointly solving for $\mathbf{z}_i$ and $\mathbf{d}$ is not—requiring the use of alternating minimization methods.

Related Problems: Sparse coding has been shown to be related or equivalent to a number of other popular feature learning methods, both supervised and unsupervised. One example is support vector machines (SVM)—a discriminative classification method that minimizes

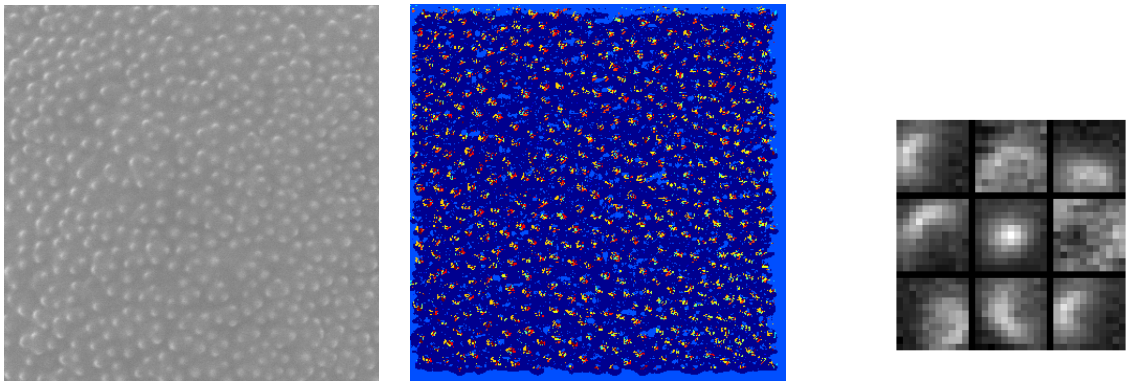
empirical risk by finding a hyperplane that maximizes the distance between two classes. If we modify Eq. 2.1 to replace the ℓ_2-norm with the standard Hilbert norm, then Girosi has shown that the solution to that problem will be the same solution as that of SVM on the same data—conditioned on the data being noiseless and having been sampled from a function that has zero mean in Hilbert space [17]. Another example is k-means clustering—a vector quantization method that partitions the data into groups by assigning each observation $\mathbf{x}$ to the group with the closest mean. If we modify Eq. 2.1 again, but this time replace the ℓ_1-regularizer with a hard constraint on the number of non-zero entries in $\mathbf{z}$ (*i.e.*, $\|\mathbf{z}\|_0 = 1$), then it is clear the two methods are the same if we define closest w.r.t.squared Euclidean distance. Coates and Ng showed this connection and used modified version of k-means for feature learning [6].

Coding: Solving for the code $\mathbf{z}$ when the dictionary (basis vectors) are known is a convex optimization problem that has been studied extensively. A survey by Yang *et al.* [68] showed coding methods generally fell into two categories: classical methods and first-order methods. Classical methods like primal-dual interior-point algorithm, which iteratively solve primal and dual variables of the problem, and homotopy methods, which exploits the characteristic of the problem that the objective function undergoes a homotopy from the reconstruction (ℓ_2) term to the sparsity (ℓ_1) term as weight of the sparsity term (β) decreases. And first-order methods like proximal-point methods, parallel coordinate descent, and convex cone solves, which all explicitly make use of the subdifferential structure of the ℓ_1 term.

Dictionary Learning: Learning the set of basis vectors is commonly referred to as sparse dictionary learning in the literature, with the basis vectors being equated to atoms (or words) in a dictionary. The dictionary learning problem shares many similarities with principal component analysis, where in fact the principal components of the data can be used as the basis vectors from which to reconstruct the data from. The above formulation differs as it does

not learn an orthogonal set of basis vectors that will decorrelate the variables—allowing us to learn undercomplete, complete, or overcomplete dictionaries. Undercomplete dictionaries are strongly related to dimensionality reduction methods as the resulting representation is reduced. Overcomplete dictionaries provide a richer representation and is commonly used in tasks like classification.

2.2 Convolutional Sparse Coding



(a) (b) (c)

Figure 2.1: (a) Example *A. odoratum* grass pollen grain image (SEM, 6000x). (b) Visualization of sparse coding of pollen texture. Pseudo-color indicates dictionary element with the largest magnitude coefficient (c) Corresponding dictionary learned from images of pollen texture using convolutional sparse coding with $\beta = 1$

The traditional sparse coding formulation reconstructs input vectors of the same length as the basis vectors, for images, this generally means reconstructing image patches rather than whole images. However, each input is treated as an independent observation—an assumption which is clearly not true as image patches in the same local neighborhood are shifted versions of one another. This leads to dictionaries that contain shifted version of the same atom, which reduces the richness of the representation [29].

Convolutional sparse coding remedies this shortcoming by modeling shift invariance directly.

This yields more parsimonious coding schemes but increases complexity by coupling the coding problem for neighboring image patches where they overlap. Here we describe an optimization approach which exploits the separability of convolution across bands in the frequency domain in order to make dictionary learning efficient. Our presentation follows the notation in the paper of [3] but includes additional implementation details and corrects some inaccuracies and typos.

2.2.1 Convolutional coding in the frequency domain

The objective for convolutional sparse coding is

$$\begin{aligned} \arg\min_{\mathbf{d},\mathbf{z}} \frac{1}{2}\|\mathbf{x} - \sum_{k=1}^{K} \mathbf{d}_k * \mathbf{z}_k\|_2^2 + \beta \sum_{k=1}^{K} \|\mathbf{z}_k\|_1 \\ \text{s.t. } \|\mathbf{d}_k\|_2^2 \leq 1 \quad \forall k = 1, \ldots, K, \end{aligned} \tag{2.2}$$

where $\mathbf{d}_k \in \mathbb{R}^M$ is the k-th filter, $\mathbf{z}_k \in \mathbb{R}^D$ is the corresponding sparse feature map, and $\mathbf{x} \in \mathbb{R}^{D-M+1}$ is an image. (Note that $M \ll D$.) For simplicity, we discuss using 1-dimensional image signals, but this generalizes to 2-dimensional image signals as well. It is well known that convolution corresponds to element-wise multiplication in the frequency domain, so it is more efficient to rewrite this objective in the frequency domain. Unfortunately this cannot be done due to the ℓ_1 penalty term, which is not rotation invariant. One way to work around this problem is by introducing two auxiliary variables $\mathbf{s}$ and $\mathbf{t}$ so that we can optimize part of the problem in the frequency domain and another part in the spatial domain. The objective

can be rewritten as

$$\begin{aligned}
\arg\min_{\hat{\mathbf{d}},\mathbf{s},\hat{\mathbf{z}},\mathbf{t}} \frac{1}{2D}\|\hat{\mathbf{x}} - \sum_{k=1}^{K} \hat{\mathbf{d}}_k \odot \hat{\mathbf{z}}_k\|_2^2 + \beta \sum_{k=1}^{K} \|\mathbf{t}_k\|_1 \\
\text{s.t. } \|\mathbf{s}_k\|_2^2 \leq 1 \quad \forall k = 1, \ldots, K \\
\mathbf{s}_k = \mathbf{F}^{-1}\hat{\mathbf{d}}_k \quad \forall k = 1, \ldots, K \\
\mathbf{t}_k = \mathbf{F}^{-1}\hat{\mathbf{z}}_k \quad \forall k = 1, \ldots, K \\
\mathbf{s}_k = \mathbf{C}\mathbf{s}_k \quad \forall k = 1, \ldots, K,
\end{aligned} \tag{2.3}$$

where $\hat{\mathbf{x}}, \hat{\mathbf{d}}_k, \hat{\mathbf{z}}_k \in \mathbb{C}^D$, $\mathbf{s}_k, \mathbf{t}_k \in \mathbb{R}^D$, and $\mathbf{F}^{-1}$ is the inverse discrete Fourier transform matrix. The original variables $\mathbf{d}$ and $\mathbf{z}$ are moved to the frequency domain, and are now denoted as $\hat{\mathbf{d}}$ and $\hat{\mathbf{z}}$, while the auxiliary variables $\mathbf{s}$ and $\mathbf{t}$ take their place in the spatial domain and constraints are added to couple them together. The careful observer may also noticed that although originally $\mathbf{d}_k$ was M dimensional, $\mathbf{s}_k$ is now D dimensional. The mapping matrix $\mathbf{C}$ zeros out any entries of $\mathbf{s}_k$ which are outside the desired spatial support so the final constraint assures the learned dictionary elements have small spatial extent.

Notation: $\mathbf{F}$ is the discrete Fourier transform matrix and $\mathbf{F}^{-1}$ is the inverse DFT matrix defined as $\mathbf{F}^{-1} = \frac{1}{D}\mathbf{F}^{\dagger}$. Since the transform matrix $\mathbf{F}$ is orthogonal, the squared ℓ_2-norm in the spatial and frequency domains are equivalent up to a scale factor (i.e., $D\|\mathbf{s}\|_2^2 = \|\hat{\mathbf{s}}\|_2^2$). The matrix $\mathbf{C}$ is the mapping matrix that truncates a vector $\mathbb{R}^D \rightarrow \mathbb{R}^M$ and pads the truncation with zeros back to $\mathbb{R}^D$. This is can be defined as the product of two matrices $\mathbf{C} = \mathbf{P}\mathbf{T}$, where $\mathbf{P}$ is our padding matrix (a $D \times M$ identity matrix) and $\mathbf{T}$ is our truncation matrix (a $M \times D$ identity matrix). For the vectors $\mathbf{d}$, $\mathbf{s}$, $\mathbf{z}$, and $\mathbf{t}$ where the subscript is dropped, we refer to a super vector with K components stacked together (e.g. $\mathbf{s} = [\mathbf{s}_1^{\intercal}, \ldots, \mathbf{s}_K^{\intercal}]^{\intercal}$). The same applies for the vectors in the frequency domain, $\hat{\mathbf{s}} = [(\mathbf{F}\mathbf{s}_1)^{\dagger}, \ldots, (\mathbf{F}\mathbf{s}_K)^{\dagger}]^{\dagger}$.

2.2.2 Optimization by ADMM

To solve this problem with variables in both the frequency and spatial domains, the Alternating Direction Method of Multipliers (ADMM) approach is used. We begin by writing the augmented Lagrangian that incorporates the equality constraints coupling the auxiliary variables $\mathbf{s}, \mathbf{t}$ to the original variables $\mathbf{d}, \mathbf{z}$.

$$\begin{aligned}\mathcal{L}(\hat{\mathbf{d}}, \mathbf{s}, \hat{\mathbf{z}}, \mathbf{t}, \boldsymbol{\lambda}_{\mathbf{s}}, \boldsymbol{\lambda}_{\mathbf{t}}) = \frac{1}{2D}\|\hat{\mathbf{x}} - \sum_{k=1}^{K} \hat{\mathbf{d}}_k \odot \hat{\mathbf{z}}_k\|_2^2 + \beta \sum_{k=1}^{K} \|\mathbf{t}_k\|_1 \\ + \sum_{k=1}^{K} \frac{\mu_{\mathbf{s}}}{2}\|\mathbf{F}^{-1}\hat{\mathbf{d}}_k - \mathbf{s}_k\|_2^2 + \sum_{k=1}^{K} \boldsymbol{\lambda}_{\mathbf{s}k}^{\intercal}(\mathbf{F}^{-1}\hat{\mathbf{d}}_k - \mathbf{s}_k) \\ + \sum_{k=1}^{K} \frac{\mu_{\mathbf{t}}}{2}\|\mathbf{F}^{-1}\hat{\mathbf{z}}_k - \mathbf{t}_k\|_2^2 + \sum_{k=1}^{K} \boldsymbol{\lambda}_{\mathbf{t}k}^{\intercal}(\mathbf{F}^{-1}\hat{\mathbf{z}}_k - \mathbf{t}_k) \\ \text{s.t. } \|\mathbf{s}_k\|_2^2 \leq 1 \quad \forall k = 1, \ldots, K \\ \mathbf{s}_k = \mathbf{C}\mathbf{s}_k \quad \forall k = 1, \ldots, K,\end{aligned} \tag{2.4}$$

where $\boldsymbol{\lambda}_{\mathbf{s}k}$, $\mu_{\mathbf{s}}$, $\boldsymbol{\lambda}_{\mathbf{t}k}$, and $\mu_{\mathbf{t}}$ are the Lagrange multipliers associated with $\mathbf{s}$ and $\mathbf{t}$ respectively.

Subproblem z:

$$\begin{aligned}\hat{\mathbf{z}}^* &= \arg\min_{\hat{\mathbf{z}}} \mathcal{L}(\hat{\mathbf{z}}; \hat{\mathbf{d}}, \mathbf{s}, \mathbf{t}, \boldsymbol{\lambda}_{\mathbf{s}}, \boldsymbol{\lambda}_{\mathbf{t}}) \\ &= \arg\min_{\hat{\mathbf{z}}} \frac{1}{2D}\|\hat{\mathbf{x}} - \sum_{k=1}^{K} \hat{\mathbf{d}}_k \odot \hat{\mathbf{z}}_k\|_2^2 + \sum_{k=1}^{K} \frac{\mu_{\mathbf{t}}}{2}\|\mathbf{F}^{-1}\hat{\mathbf{z}}_k - \mathbf{t}_k\|_2^2 + \sum_{k=1}^{K} \boldsymbol{\lambda}_{\mathbf{t}k}^{\intercal}(\mathbf{F}^{-1}\hat{\mathbf{z}}_k - \mathbf{t}_k) \\ &= \arg\min_{\hat{\mathbf{z}}} \frac{1}{2D}\|\hat{\mathbf{x}} - \hat{\mathbf{D}}\hat{\mathbf{z}}\|_2^2 + \frac{\mu_{\mathbf{t}}}{2D}\|\hat{\mathbf{z}} - \hat{\mathbf{t}}\|_2^2 + \frac{1}{D}\hat{\boldsymbol{\lambda}}_{\mathbf{t}}^{\dagger}(\hat{\mathbf{z}} - \hat{\mathbf{t}}) \\ &= (\hat{\mathbf{D}}^{\dagger}\hat{\mathbf{D}} + \mu_{\mathbf{t}}\mathbf{I})^{-1}(\hat{\mathbf{D}}^{\dagger}\hat{\mathbf{x}} + \mu_{\mathbf{t}}\hat{\mathbf{t}} - \hat{\boldsymbol{\lambda}}_{\mathbf{t}}),\end{aligned} \tag{2.5}$$

where $\hat{\mathbf{D}} = [\text{diag}(\hat{\mathbf{d}}_1), \ldots, \text{diag}(\hat{\mathbf{d}}_K)]$ is $D \times KD$. Although the matrix $\hat{\mathbf{D}}^{\dagger}\hat{\mathbf{D}}$ is quite large, $KD \times KD$, it is sparse banded and has an efficient variable reordering such that $\hat{\mathbf{z}}^*$ can be found by solving D independent $K \times K$ linear systems (see Sec. A.2 for more details).

Subproblem t:

$$\begin{aligned}
\mathbf{t}_k^* &= \arg\min_{\mathbf{t}_k} \mathcal{L}(\mathbf{t}_k; \hat{\mathbf{d}}_k, \mathbf{s}_k, \hat{\mathbf{z}}_k, \boldsymbol{\lambda}_{\mathbf{s}k}, \boldsymbol{\lambda}_{\mathbf{t}k}) \\
&= \arg\min_{\mathbf{t}_k} \beta\|\mathbf{t}_k\|_1 + \frac{\mu_{\mathbf{t}}}{2}\|\mathbf{F}^{-1}\hat{\mathbf{z}}_k - \mathbf{t}_k\|_2^2 + \boldsymbol{\lambda}_{\mathbf{t}k}^\intercal(\mathbf{F}^{-1}\hat{\mathbf{z}}_k - \mathbf{t}_k) \\
&= \arg\min_{\mathbf{t}_k} \beta\|\mathbf{t}_k\|_1 + \frac{\mu_{\mathbf{t}}}{2}\|\mathbf{z}_k - \mathbf{t}_k\|_2^2 + \boldsymbol{\lambda}_{\mathbf{t}k}^\intercal(\mathbf{z}_k - \mathbf{t}_k)
\end{aligned} \tag{2.6}$$

Given $\mathbf{z}_k$ and $\boldsymbol{\lambda}_{\mathbf{t}k}$ in the spatial domain, it turns out that each coefficient in $\mathbf{t}_k$ is independent of all other coefficients,

$$t_{ki}^* = \arg\min_{t_{ki}} \frac{\mu_{\mathbf{t}}}{2}(z_{ki} - t_{ki})^2 + \lambda_{\mathbf{t}ki}(z_{ki} - t_{ki}) + \beta|t_{ki}|. \tag{2.7}$$

Which can be efficiently compute in closed form using the shrinkage (soft-thresholding) function,

$$t_{ki}^* = \operatorname{sign}\left(z_{ki} + \frac{\lambda_{\mathbf{t}ki}}{\mu_{\mathbf{t}}}\right) \cdot \max\left\{\left|z_{ki} + \frac{\lambda_{\mathbf{t}ki}}{\mu_{\mathbf{t}}}\right| - \frac{\beta}{\mu_{\mathbf{t}}}, 0\right\}. \tag{2.8}$$

Subproblem d:

$$\begin{aligned}
\hat{\mathbf{d}}^* &= \arg\min_{\hat{\mathbf{d}}} \mathcal{L}(\hat{\mathbf{d}}; \mathbf{s}, \hat{\mathbf{z}}, \mathbf{t}, \boldsymbol{\lambda}_{\mathbf{s}}, \boldsymbol{\lambda}_{\mathbf{t}}) \\
&= \arg\min_{\hat{\mathbf{d}}} \frac{1}{2D}\|\hat{\mathbf{x}} - \sum_{k=1}^{K} \hat{\mathbf{d}}_k \odot \hat{\mathbf{z}}_k\|_2^2 + \sum_{k=1}^{K} \frac{\mu_{\mathbf{s}}}{2}\|\mathbf{F}^{-1}\hat{\mathbf{d}}_k - \mathbf{s}_k\|_2^2 + \sum_{k=1}^{K} \boldsymbol{\lambda}_{\mathbf{s}k}^\intercal(\mathbf{F}^{-1}\hat{\mathbf{d}}_k - \mathbf{s}_k) \\
&= \arg\min_{\hat{\mathbf{d}}} \frac{1}{2D}\|\hat{\mathbf{x}} - \hat{\mathbf{Z}}\hat{\mathbf{d}}\|_2^2 + \frac{\mu_{\mathbf{s}}}{2D}\|\hat{\mathbf{d}} - \hat{\mathbf{s}}\|_2^2 + \frac{1}{D}\hat{\boldsymbol{\lambda}}_{\mathbf{s}}^\dagger(\hat{\mathbf{d}} - \hat{\mathbf{s}}) \\
&= (\hat{\mathbf{Z}}^\dagger\hat{\mathbf{Z}} + \mu_{\mathbf{s}}\mathbf{I})^{-1}(\hat{\mathbf{Z}}^\dagger\hat{\mathbf{x}} + \mu_{\mathbf{s}}\hat{\mathbf{s}} - \hat{\boldsymbol{\lambda}}_{\mathbf{s}})
\end{aligned} \tag{2.9}$$

where $\hat{\mathbf{Z}} = [\operatorname{diag}(\hat{\mathbf{z}}_1), \ldots, \operatorname{diag}(\hat{\mathbf{z}}_K)]$ is $D \times KD$. Like subproblem $\mathbf{z}$, there is an efficient variable reordering such that $\hat{\mathbf{d}}^*$ can be found by solving D independent $K \times K$ linear systems (see Sec. A.2 for more details).

Subproblem s:

$$\begin{aligned}
\mathbf{s}_k^* &= \arg\min_{\mathbf{s}_k} \mathcal{L}(\mathbf{s}_k; \hat{\mathbf{d}}_k, \hat{\mathbf{z}}_k, \mathbf{t}_k, \boldsymbol{\lambda}_{\mathbf{s}k}, \boldsymbol{\lambda}_{\mathbf{t}k}) \\
&= \arg\min_{\mathbf{s}_k} \frac{\mu_{\mathbf{s}}}{2}\|\mathbf{F}^{-1}\hat{\mathbf{d}}_k - \mathbf{s}_k\|_2^2 + \boldsymbol{\lambda}_{\mathbf{s}k}^{\intercal}(\mathbf{F}^{-1}\hat{\mathbf{d}}_k - \mathbf{s}_k) \\
&= \arg\min_{\mathbf{s}_k} \frac{\mu_{\mathbf{s}}}{2}\|\mathbf{d}_k - \mathbf{s}_k\|_2^2 + \boldsymbol{\lambda}_{\mathbf{s}k}^{\intercal}(\mathbf{d}_k - \mathbf{s}_k) \\
&\text{s.t. } \|\mathbf{s}_k\|_2^2 \leq 1 \\
&\quad \mathbf{s}_k = \mathbf{C}\mathbf{s}_k
\end{aligned} \tag{2.10}$$

An equivalent formulation is to drop the spatial support constraint on $\mathbf{s}$ and instead preserve only the M coefficients of $\mathbf{d}_k$ and $\boldsymbol{\lambda}_{\mathbf{s}k}$ associated with the small spatial support.

$$\begin{aligned}
\mathbf{s}_k^* &= \arg\min_{\mathbf{s}_k} \frac{\mu_{\mathbf{s}}}{2}\|\mathbf{C}\mathbf{d}_k - \mathbf{s}_k\|_2^2 + (\mathbf{C}\boldsymbol{\lambda}_{\mathbf{s}k})^{\intercal}(\mathbf{C}\mathbf{d}_k - \mathbf{s}_k) \\
&\text{s.t. } \|\mathbf{s}_k\|_2^2 \leq 1.
\end{aligned} \tag{2.11}$$

The coefficients of $\mathbf{s}_k$ outside the spatial support (where $\mathbf{C}\mathbf{d}_k$ is 0) are automatically driven to 0 by the ℓ_2-norm (see Sec. A.1 for details).

We can solve the unconstrained version of this problem by pulling the linear term into the quadratic, completing the square

$$\begin{aligned}
\tilde{\mathbf{s}}_k^* &= \arg\min_{\mathbf{s}_k} \frac{\mu_{\mathbf{s}}}{2}\|\mathbf{C}\mathbf{d}_k - \mathbf{s}_k\|_2^2 + (\mathbf{C}\boldsymbol{\lambda}_{\mathbf{s}k})^{\intercal}(\mathbf{C}\mathbf{d}_k - \mathbf{s}_k) \\
&= \arg\min_{\mathbf{s}_k} \frac{\mu_{\mathbf{s}}}{2}\|\bar{\mathbf{d}}_k - \mathbf{s}_k\|_2^2 + \bar{\boldsymbol{\lambda}}_{\mathbf{s}k}^{\intercal}(\bar{\mathbf{d}}_k - \mathbf{s}_k) \qquad \text{where } \bar{\mathbf{d}}_k = \mathbf{C}\mathbf{d}_k, \bar{\boldsymbol{\lambda}}_{\mathbf{s}k} = \mathbf{C}\boldsymbol{\lambda}_{\mathbf{s}k} \\
&= \arg\min_{\mathbf{s}_k} \frac{\mu_{\mathbf{s}}}{2}\|\mathbf{s}_k - \big(\bar{\mathbf{d}}_k + \frac{1}{\mu_{\mathbf{s}}}\bar{\boldsymbol{\lambda}}_{\mathbf{s}k}\big)\|_2^2 - \frac{\mu_{\mathbf{s}}}{2}\|\bar{\mathbf{d}}_k + \frac{1}{\mu_{\mathbf{s}}}\bar{\boldsymbol{\lambda}}_{\mathbf{s}k}\|_2^2 + \frac{\mu_{\mathbf{s}}}{2}\bar{\mathbf{d}}_k{}^{\intercal}\bar{\mathbf{d}}_k + \bar{\boldsymbol{\lambda}}_{\mathbf{s}k}^{\intercal}\bar{\mathbf{d}}_k \\
&= \arg\min_{\mathbf{s}_k} \frac{\mu_{\mathbf{s}}}{2}\|\mathbf{s}_k - \big(\bar{\mathbf{d}}_k + \frac{1}{\mu_{\mathbf{s}}}\bar{\boldsymbol{\lambda}}_{\mathbf{s}k}\big)\|_2^2
\end{aligned}$$

The optimal solution to the unconstrained problem (denoted by $\tilde{\mathbf{s}}_k$) is thus given in closed

form by

$$\tilde{\mathbf{s}}_k = \mathbf{C}\mathbf{d}_k + \frac{1}{\mu_{\mathbf{s}}}\mathbf{C}\boldsymbol{\lambda}_{\mathbf{s}k}, \tag{2.12}$$

Since this objective is isotropic, we can simply enforce the norm constraint by projecting the unconstrained solution back on to the constraint set:

$$\mathbf{s}_k^* = \begin{cases} \|\tilde{\mathbf{s}}_k\|_2^{-1}\tilde{\mathbf{s}}_k & \text{if } \|\tilde{\mathbf{s}}_k\|_2^2 \geq 1 \\ \tilde{\mathbf{s}}_k & \text{otherwise} \end{cases}. \tag{2.13}$$

Lagrange Multiplier Update:

$$\boldsymbol{\lambda}_{\mathbf{s}}^{(i+1)} = \boldsymbol{\lambda}_{\mathbf{s}}^{(i)} + \mu_{\mathbf{s}}(\mathbf{d}^{(i+1)} - \mathbf{s}^{(i+1)}) \tag{2.14}$$

$$\boldsymbol{\lambda}_{\mathbf{t}}^{(i+1)} = \boldsymbol{\lambda}_{\mathbf{t}}^{(i)} + \mu_{\mathbf{t}}(\mathbf{z}^{(i+1)} - \mathbf{t}^{(i+1)}) \tag{2.15}$$

Penalty Update:

$$\mu^{(i+1)} = \begin{cases} \tau\mu^{(i)} & \text{if } \mu^{(i)} < \mu_{\max} \\ \mu^{(i)} & \text{otherwise} \end{cases} \tag{2.16}$$

With the solutions to all the subproblems, we can solve for Eq. 2.3 with an iterative algorithm (Alg. 1) that alternates between updating the dictionary, codes, Lagrange multipliers, and penalties.

Algorithm 1 Convolutional Sparse Coding using Fourier Subproblems and ADMM

Initialize $\mathbf{s}^{(0)}, \mathbf{z}^{(0)}, \mathbf{t}^{(0)}, \boldsymbol{\lambda}_{\mathbf{s}}^{(0)}, \boldsymbol{\lambda}_{\mathbf{t}}^{(0)}, \mu_{\mathbf{s}}^{(0)}, \mu_{\mathbf{t}}^{(0)}$

Perform FFT $\mathbf{s}^{(0)}, \mathbf{z}^{(0)}, \mathbf{t}^{(0)}, \boldsymbol{\lambda}_{\mathbf{s}}^{(0)}, \boldsymbol{\lambda}_{\mathbf{t}}^{(0)} \rightarrow \hat{\mathbf{s}}^{(0)}, \hat{\mathbf{z}}^{(0)}, \hat{\mathbf{t}}^{(0)}, \hat{\boldsymbol{\lambda}}_{\mathbf{s}}^{(0)}, \hat{\boldsymbol{\lambda}}_{\mathbf{t}}^{(0)}$

$i = 0$

repeat

    dictionary update:

    Solve for $\hat{\mathbf{d}}^{(i+1)}$ given $\hat{\mathbf{s}}^{(i)}, \hat{\mathbf{z}}^{(i)}, \hat{\boldsymbol{\lambda}}_{\mathbf{s}}^{(i)}$ using Eq. 2.9

    Perform inverse FFT $\hat{\mathbf{d}}^{(i+1)} \rightarrow \mathbf{d}^{(i+1)}$

    Solve for $\tilde{\mathbf{s}}^{(i+1)}$ given $\mathbf{d}^{(i+1)}$ using Eq. 2.12

    Solve for $\mathbf{s}^{(i+1)}$ by projecting $\tilde{\mathbf{s}}^{(i+1)}$ using Eq. 2.13

    code update:

    Solve for $\hat{\mathbf{z}}^{(i+1)}$ given $\hat{\mathbf{t}}^{(i)}, \hat{\boldsymbol{\lambda}}_{\mathbf{t}}^{(i)}, \hat{\mathbf{d}}^{(i+1)}$ using Eq. 2.5

    Perform inverse FFT $\hat{\mathbf{z}}^{(i+1)} \rightarrow \mathbf{z}^{(i+1)}$

    Solve for $\mathbf{t}^{(i+1)}$ given $\mathbf{z}^{(i+1)}, \boldsymbol{\lambda}_{\mathbf{t}}^{(i)}$ using Eq. 2.6

    Update Lagrange multiplier vectors using Eq. 2.14 and Eq. 2.15

    Update penalties using Eq. 2.16

    Perform FFT $\mathbf{s}, \mathbf{t}, \boldsymbol{\lambda}_{\mathbf{s}}, \boldsymbol{\lambda}_{\mathbf{t}} \rightarrow \hat{\mathbf{s}}, \hat{\mathbf{t}}, \hat{\boldsymbol{\lambda}}_{\mathbf{s}}, \hat{\boldsymbol{\lambda}}_{\mathbf{t}}$

    $i = i + 1$

until $\hat{\mathbf{d}}, \mathbf{s}, \hat{\mathbf{z}}, \mathbf{t}$ has converged

2.2.3 Effect of ADMM parameters on optimization

This section aims to provide some intuition regarding the hyper-parameters of FCSC optimization, specifically we want to understand how the hyper-parameters interact with one another in terms of the convergence rate and the objective value. In all our experiments, we use images that are contrast normalized. The convergence criteria is that we've reach a feasible solution (i.e., $\mathbf{s} = \mathbf{F}^{-1}\hat{\mathbf{d}}$ and $\mathbf{t} = \mathbf{F}^{-1}\hat{\mathbf{z}}$).

The FCSC algorithm has three hyper-parameters, $\mu_{\mathbf{s}}$, $\mu_{\mathbf{t}}$, and β. $\mu_{\mathbf{s}}$ can be thought of as the parameter that control the convergence rate of the dictionary learning, as it is the step-size at which we move towards a feasible solution. Likewise, $\mu_{\mathbf{t}}$ is the same for the coding. The hyper-parameter β in our original objective controls the sparsity of the codes found.

Effect of step-size parameters on coding

First we look at the coding hyper-parameters; using a pre-trained dictionary, we vary β and $\mu_{\mathbf{t}}$. Since β controls the sparsity of the codes, we expect (and the results show) that as it increases the reconstruction error would also increase. Perhaps not surprisingly, the same behavior is observed w.r.t.the objective value. As β increases the codes become more sparse to the point where they're all zeros. At that point the objective value is just the reconstruction error with a vector of zeros. We see that both the reconstruction error and the objective value at convergence increase slowly as $\mu_{\mathbf{t}}$ increases. In another experiments (not shown) we have seen the objective value be roughly the same for all values of $\mu_{\mathbf{t}}$ and $\beta = 0$, when a minimum number of iterations is enforced (in those experiments it was 10). These iterations are necessary to overcome the "memory" of the initial value of $\mathbf{t}$ in the ADMM iteration.

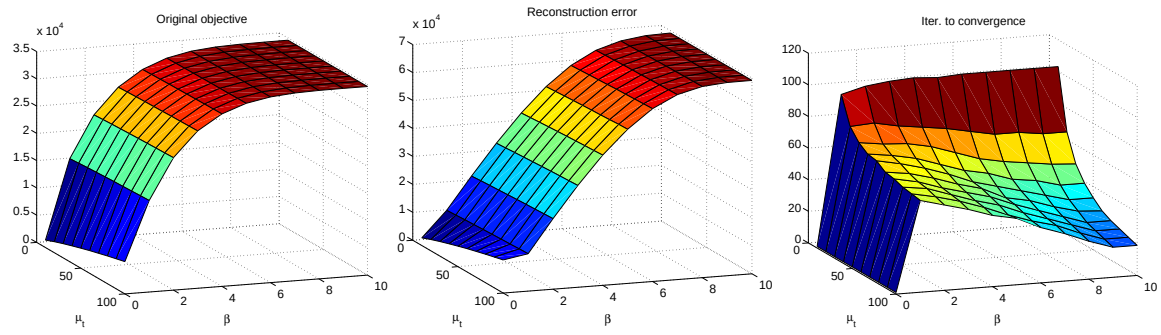


Figure 2.2: Effect of $\mu_{\mathbf{t}}, \beta$ on coding with fixed dictionary.

In terms of the convergence rate, for all values of β we generally see that the number of iterations required before convergence decreases as $\mu_{\mathbf{t}}$ increases. This is also to be expected

as we previously described $\mu_{\mathbf{t}}$ as the step-size towards a feasible solution. Ignoring $\beta = 0$, we generally observe that the number of iterations goes down as β increases for any particular value of $\mu_{\mathbf{t}}$. This is a bit surprising, but can possibly be explained by considering how sparsity is induced. We know that the shrinkage function is used to solve the $\mathbf{t}$-subproblem, and that at each iteration the codes are shrunk by $\frac{\beta}{\mu_{\mathbf{t}}}$ towards zero. If β is large enough, then $\mu_{\mathbf{t}}$ does not grow enough at the early iterations such that additional coefficients of $\mathbf{z}$ are not zeroed out. This in turn magnifies the coefficients of $\boldsymbol{\lambda}_{\mathbf{t}}$ to make $\mathbf{z}$ more like $\mathbf{t}$, thus leading to faster convergence.

Effect of step-size and sparsity parameters on dictionary learning

In the second experiment, we fix the step-size $\mu_{\mathbf{s}}$ and simultaneously learn the dictionary and the codes. Here the results are quite different from before. We see that $\mu_{\mathbf{t}}$ must be strictly greater than β for FCSC to learn anything meaningful. When $\beta = 1$ both the reconstruction error and the objective value are largely unchanged w.r.t. $\mu_{\mathbf{t}}$. Finally in terms of convergence, we continue to observe that as $\mu_{\mathbf{t}}$ increases the number of iterations to convergence decreases for all values of β, consistent with our first experiment.

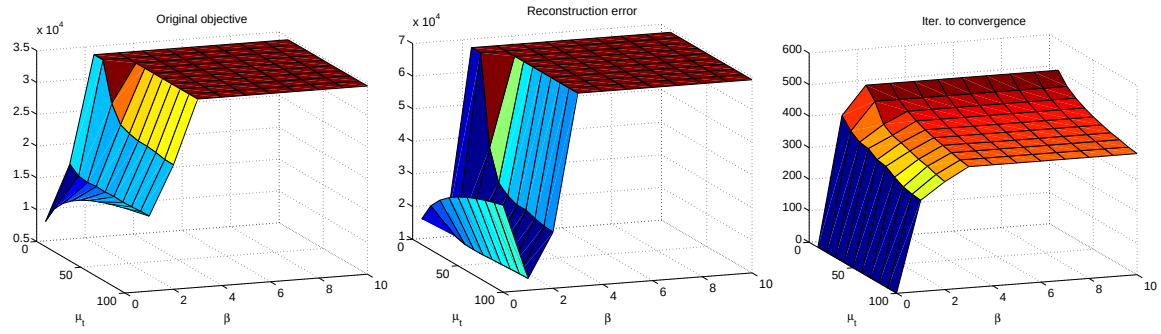


Figure 2.3: Effect of $\mu_{\mathbf{t}}, \beta$ on joint coding and dictionary learning ($\mu_{\mathbf{s}} = 515$).

Effect of step-size on runtime and accuracy

In the third experiment, we fix the β hyper-parameter equal to 1 and look at how $\mu_{\mathbf{s}}$ and $\mu_{\mathbf{t}}$ interact. With β fixed, we see that the objective values increase as $\mu_{\mathbf{t}}$ and $\mu_{\mathbf{s}}$ increase in the domain $\mu_{\mathbf{s}} > 2000$ and $\mu_{\mathbf{t}} > 5$. This seems to be a result of the optimization enforcing constraints "too quickly". For small values of $\mu_{\mathbf{t}}$ and $\mu_{\mathbf{s}}$ (not shown) we also see erratic behavior. If we ignore the region where $\mu_{\mathbf{t}}$ is less than 5, we see that both the objective and the reconstruction error are smooth w.r.t. the hyper-parameters. This is quite nice, as we can choose a computational trade-off between lowering the objective value and the number of iterations to convergence.

We may also ask whether the optimization produces qualitatively "good" filters. A good learned filter shouldn't resemble the initialization and in general we expect to see Gabor-like oriented edge filters. After examining some learned filters that were subjectively good/bad, we defined goodness as having a standard deviation greater than 0.7. Much higher than the flat initialization which has a standard deviation of 0.01 and other bad looking filters that had standard deviation less than 0.6 in the cases we examined. While not a perfect measure by any means, we can see in the next set of figures that the reconstruction error is indeed correlated with the goodness measure.

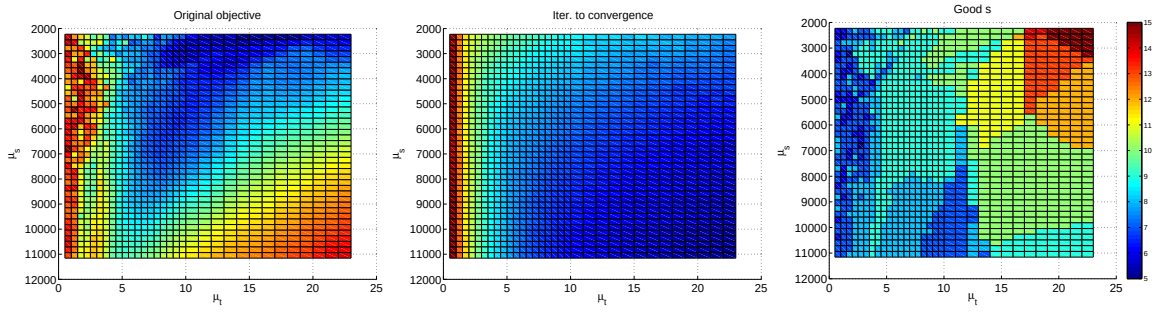


Figure 2.4: Joint effect of $\mu_{\mathbf{t}}, \mu_{\mathbf{s}}$ on solution quality and run time ($\beta = 1$)

Chapter 3

Energy-Based Spherical Sparse Coding

In this chapter we address a major limitation with using sparse coding as a feature extractor by proposing a novel formulation that uses inner product instead of least squares, thereby avoiding the need for iterative optimization. And with all the terms in the proposed formulation being linear, we are able to compute the sparse codes in a feed-forward way. We then extend the proposed formulation of sparse coding to a supervised method that is inspired by biased competition theory, where the top-down class label biases the bottom-up features that are extracted–giving us class-specific features that can be used to reconstruct a version of the input that maximizes the score of the hypothesized class.

3.1 Introduction

Sparse coding has been widely studied as a representation for images, audio and other vectorial data. This highly successful method that has found its way into many applications,

from signal compression and denoising [11, 13] to image classification [66], to modeling neuronal receptive fields in visual cortex [47]. Since its introduction, subsequent works have brought sparse coding into the supervised learning setting by introducing classification loss terms to the original formulation to encourage features that are not only able to reconstruct the original signal but are also discriminative [27, 69, 74, 26, 78, 76].

While supervised sparse coding methods have been shown to find more discriminative features leading to improved classification performance over their unsupervised counterparts, they have received much less attention in recent years and have been eclipsed by simpler feed-forward architectures.

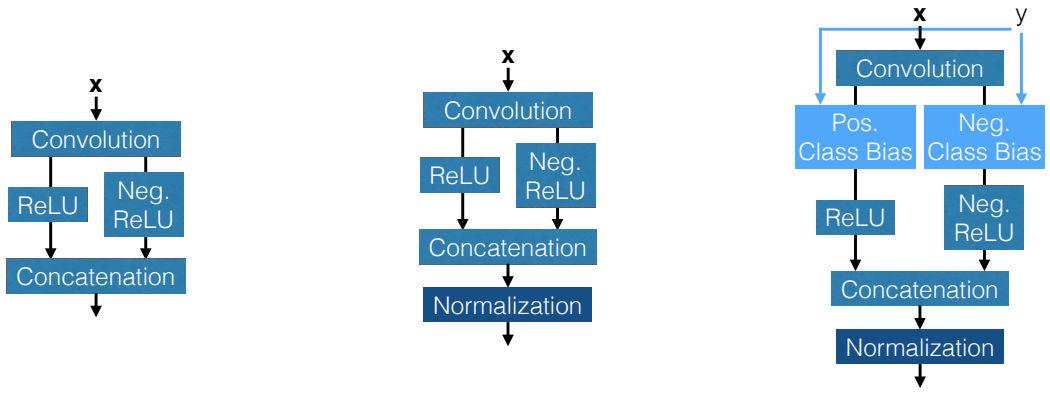
This is in part because sparse coding is computationally expensive. Convex formulations of sparse coding typically consist of a minimization problem over an objective that includes a least-squares (LSQ) reconstruction error term plus a sparsity inducing regularizer.

Because there is no closed-form solution to this formulation, various iterative optimization techniques are generally used to find a solution [74, 3, 68, 22]. In applications where an approximate solution suffices, there is work that learns non-linear predictors to estimate sparse codes rather than solve the objective more directly [19]. The computational overhead for iterative schemes becomes quite significant when training discriminative models due to the demand of processing many training examples necessary for good performance, and so sparse coding has fallen out of favor by not being able to keep up with simpler non-iterative coding methods.

In this chapter we introduce an alternate formulation of sparse coding using unit length codes and a reconstruction loss based on the cosine similarity. Optimal sparse codes in this model can be computed in a non-iterative fashion and the coding objective lends itself naturally to embedding in a discriminative, energy-based classifier which we term *energy-based spherical sparse coding (EB-SSC)*. This bi-directional coding method incorporates both top-down and

bottom-up information where the features representation depends on both a hypothesized class label and the input signal. Like [4], our motivation for bi-directional coding comes from the "Biased Competition Theory," which suggests that visual processing can be biased by other mental processes (*e.g.*, top-down influence) to prioritize certain features that are most relevant to current task. Fig. 3.1 illustrates the flow of computation used by our SSC and EB-SSC building blocks compared to a standard feed-forward layer.

Our energy based approach for combining top-down and bottom-up information is closely tied to the ideas of [38, 26, 76, 43]—although the model details are substantially different (*e.g.*, [26] and [76] use sigmoid non-linearities while [43] use separate representations for top-down and bottom-up information). The energy function of [38] is also similar but includes an extra classification term and is trained as a restricted Boltzmann machine.



(a) CReLU (b) SSC (c) EB-SSC

Figure 3.1: Building blocks for coding networks explored in this paper. Our coding model uses non-linearities that are closely related to the standard ReLU activation function. (a) Keeping both positive and negative activations provides a baseline feed-forward model termed concatenated ReLU (CReLU). (b) Our spherical sparse coding layer has a similar structure but with an extra bias and normalization step. Our proposed energy-based model uses (c) energy-based spherical sparse coding (EB-SSC) blocks that produces sparse activations which are not only positive and negative, but are class-specific. These blocks can be stacked to build deeper architectures.

3.1.1 Notation

Matrices are denoted as uppercase bold (*e.g.*, $\mathbf{A}$), vectors are lowercase bold (*e.g.*, $\mathbf{a}$), and scalars are lowercase (*e.g.*, a). We denote the transpose operator with $^\intercal$, the element-wise multiplication operator with $\odot$, the convolution operator with $*$, and the cross-correlation operator with $\star$. For vectors where we dropped the subscript k (*e.g.*, $\mathbf{d}$ and $\mathbf{z}$), we refer to a super vector with K components stacked together (*e.g.*, $\mathbf{z} = [\mathbf{z}_1^\intercal, \ldots, \mathbf{z}_K^\intercal]^\intercal$).

3.2 Energy-Based Spherical Sparse Coding

Energy-based models capture dependencies between variables using an energy function that measures the compatibility of the configuration of variables [40]. To measure the compatibility between the top-down and bottom-up information, we define the energy function of EB-SSC to be the sum of bottom-up coding term and a top-down classification term:

$$E(\mathbf{x}, y, \mathbf{z}) = E_{code}(\mathbf{x}, \mathbf{z}) + E_{class}(y, \mathbf{z}). \tag{3.1}$$

The bottom-up information (input signal $\mathbf{x}$) and the top-down information (class label y) are tied together by a latent feature map $\mathbf{z}$.

3.2.1 Bottom-Up Reconstruction

To measure the compatibility between the input signal $\mathbf{x}$ and the latent feature maps $\mathbf{z}$, we introduce a novel variant of sparse coding that is amenable to efficient feed-forward optimization. While the idea behind this variant can be applied to either patch-based or convolutional sparse coding, we specifically use the convolutional variant that shares the burden of coding an image among nearby overlapping dictionary elements. Using such

a shift-invariant approach avoids the need to learn dictionary elements which are simply translated copies of each other, freeing up resources to discover more diverse and specific filters (see [29]).

Convolutional sparse coding (CSC) attempts to find a set of dictionary elements $\{\mathbf{d}_1, \ldots, \mathbf{d}_K\}$ and corresponding sparse codes $\{\mathbf{z}_1, \ldots, \mathbf{z}_K\}$ so that the resulting reconstruction, $\mathbf{r} = \sum_{k=1}^{K} \mathbf{d}_k * \mathbf{z}_k$ accurately represents the input signal $\mathbf{x}$. This is traditionally framed as a least-squares minimization with a sparsity inducing prior on $\mathbf{z}$:

$$\arg\min_{\mathbf{z}} \|\mathbf{x} - \sum_{k=1}^{K} \mathbf{d}_k * \mathbf{z}_k\|_2^2 + \beta\|\mathbf{z}\|_1. \tag{3.2}$$

Unlike standard feed-forward CNN models that convolve the input signal $\mathbf{x}$ with the filters, this energy function corresponds to a generative model where the latent feature maps $\{\mathbf{z}_1, \ldots, \mathbf{z}_K\}$ are convolved with the filters and compared to the input signal [3, 22, 74].

To motivate our novel variant of CSC, consider expanding the squared reconstruction error $\|\mathbf{x} - \mathbf{r}\|_2^2 = \|\mathbf{x}\|_2^2 - 2\mathbf{x}^\intercal\mathbf{r} + \|\mathbf{r}\|_2^2$. If we constrain the reconstruction $\mathbf{r}$ to have unit norm, the reconstruction error depends entirely on the inner product between $\mathbf{x}$ and $\mathbf{r}$ and is equivalent to the cosine similarity (up to additive and multiplicative constants). This suggests the closely related *unit-length* reconstruction problem:

$$\begin{aligned} \arg\max_{\mathbf{z}} \; & \mathbf{x}^\intercal\big(\sum_{k=1}^{K} \mathbf{d}_k * \mathbf{z}_k\big) - \beta\|\mathbf{z}\|_1 \\ \text{s.t.} \quad & \big\|\sum_{k=1}^{K} \mathbf{d}_k * \mathbf{z}_k\big\|_2 \leq 1 \end{aligned} \tag{3.3}$$

In Sec. B.1 we show that, given an optimal unit length reconstruction $\bar{\mathbf{r}}^*$ with corresponding codes $\bar{\mathbf{z}}^*$, the solution to the least squares reconstruction problem (Eq. 3.2) can be computed by a simple scaling $\mathbf{r}^* = (\mathbf{x}^\intercal\bar{\mathbf{r}}^* - \frac{\beta}{2}\|\bar{\mathbf{z}}^*\|_1)\bar{\mathbf{r}}^*$.

The unit-length reconstruction problem is no easier than the original least-squares optimiza-

tion due to the constraint on the reconstruction which couples the codes for different filters. Instead consider a simplified constraint on $\mathbf{z}$ which we refer to as *spherical sparse coding (SSC)*:

$$\arg\max_{\|\mathbf{z}\|_2 \leq 1} E_{code}(\mathbf{x}, \mathbf{z}) \;=\; \arg\max_{\|\mathbf{z}\|_2 \leq 1} \mathbf{x}^\intercal \big(\sum_{k=1}^{K} \mathbf{d}_k * \mathbf{z}_k \big) - \beta \|\mathbf{z}\|_1. \tag{3.4}$$

In 3.2.3 below, we show that the solution to this problem can be found very efficiently without requiring iterative optimization.

This problem is a relaxation of convolutional sparse coding since it ignores non-orthogonal interactions between the dictionary elements[1]. Alternately, assuming unit norm dictionary elements, the code norm constraint can be used to upper-bound the reconstruction length. We have by the triangle and Young's inequality that:

$$\big\| \sum_k \mathbf{d}_k * \mathbf{z}_k \big\|_2 \;\leq\; \sum_k \|\mathbf{d}_k * \mathbf{z}_k\|_2 \;\leq\; \sum_k \|\mathbf{d}_k\|_1 \|\mathbf{z}_k\|_1 \;\leq\; D \sum_k \|\mathbf{z}_k\|_2 \tag{3.5}$$

where the factor D is the dimension of $\mathbf{z}_k$ and arises from switching from the 1-norm to the 2-norm. Since $D \sum_k \|\mathbf{z}_k\|_2 \leq 1$ is a tighter constraint we have

$$\max_{\|\sum_k \mathbf{d}_k * \mathbf{z}_k\|_2 \leq 1} E_{code}(\mathbf{x}, \mathbf{z}) \;\geq\; \max_{\sum_k \|\mathbf{z}_k\|_2 \leq \frac{1}{D}} E_{code}(\mathbf{x}, \mathbf{z}) \tag{3.6}$$

However, this relaxation is very loose, primarily due to the triangle inequality. Except in special cases (*e.g.*, if the dictionary elements have disjoint spectra) the SSC codes will be quite different from the standard least-squares reconstruction.

[1] We note that our formulation is also closely related to the dynamical model suggested by [55], but without the dictionary-dependent lateral inhibition between feature maps. Lateral inhibition can solve the unit-length reconstruction formulation of standard sparse coding but requires iterative optimization.

3.2.2 Top-Down Classification

To measure the compatibility between the class label y and the latent feature maps $\mathbf{z}$, we use a set of one-vs-all linear classifiers. To provide more flexibility, we generalize this by splitting the code vector into positive and negative components:

$$\mathbf{z}_k = \mathbf{z}_k^+ + \mathbf{z}_k^- \qquad \mathbf{z}_k^+ \geq 0 \qquad \mathbf{z}_k^- \leq 0$$

and allow the linear classifier to operate on each component separately. We express the classifier score for a hypothesized class label y by:

$$E_{class}(y, \mathbf{z}) = \sum_{k=1}^{K} \mathbf{w}_y^{+\intercal} \mathbf{z}_k^+ + \sum_{k=1}^{K} \mathbf{w}_y^{-\intercal} \mathbf{z}_k^-. \tag{3.7}$$

The classifier thus is parameterized by a pair of weight vectors ($\mathbf{w}_{yk}^+$ and $\mathbf{w}_{yk}^-$) for each class label y and k-th channel of the latent feature map.

This splitting, sometimes referred to as full-wave rectification, is useful since a dictionary element and its negative do not necessarily have opposite visual semantics. This splitting also allows the classifier the flexibility to assign distinct meanings or alternately be completely invariant to contrast reversal depending on the problem domain. For example, [58] found CNN models with ReLU non-linearities which discard the negative activations tend to learn pairs of filters which are related by negation. Keeping both positive and negative responses allowed them to halve the number of dictionary elements.

We note that it is also straightforward to introduce spatial average pooling prior to classification by introducing a fixed linear operator $\mathbf{P}$ used to pool the codes (e.g., $\mathbf{w}_y^{+\intercal} \mathbf{P} \mathbf{z}_k^+$). This is motivated by a variety of hand-engineered feature extractors and sparse coding models, such as [53], which use spatially pooled histograms of sparse codes for classification. This fixed pooling can be viewed as a form of regularization on the linear classifier which en-

forces shared weights over spatial blocks of the latent feature map. Splitting is also quite important to prevent information loss when performing additive pooling since positive and negative components of $\mathbf{z}_k$ can cancel each other out.

3.2.3 Coding

Bottom-up reconstruction and top-down classification each provide half of the story, coupled by the latent feature maps. For a given input $\mathbf{x}$ and hypothesized class y, we would like to find the optimal activations $\mathbf{z}$ that maximize the joint energy function $E(\mathbf{x}, y, \mathbf{z})$. This requires solving the following optimization:

$$\underset{\|\mathbf{z}\|_2 \leq 1}{\arg\max}\ \mathbf{x}^\intercal \big(\sum_{k=1}^{K} \mathbf{d}_k * \mathbf{z}_k \big) - \beta \|\mathbf{z}\|_1 + \sum_{k=1}^{K} \mathbf{w}_{yk}^{+\intercal} \mathbf{z}_k^+ + \sum_{k=1}^{K} \mathbf{w}_{yk}^{-\intercal} \mathbf{z}_k^-, \tag{3.8}$$

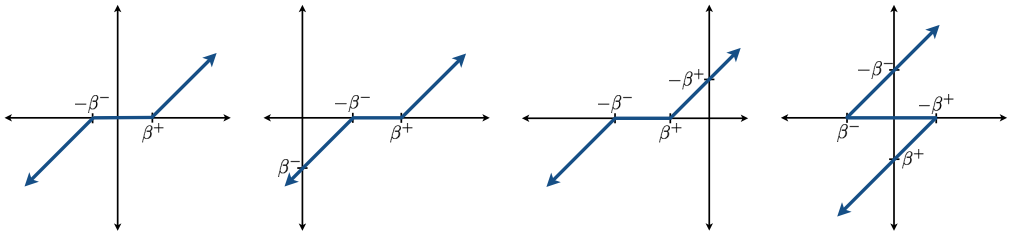
where $\mathbf{x} \in \mathbb{R}^D$ is an image and $y \in \mathcal{Y}$ is a class hypothesis. $\mathbf{z}_k \in \mathbb{R}^F$ is the k-th component latent variable being inferred; $\mathbf{z}_k^+$ and $\mathbf{z}_k^-$ are the positive and negative coefficients of $\mathbf{z}_k$, such that $\mathbf{z}_k = \mathbf{z}_k^+ + \mathbf{z}_k^-$. The parameters $\mathbf{d}_k \in \mathbb{R}^M$, $\mathbf{w}_{yk}^+ \in \mathbb{R}^F$, and $\mathbf{w}_{yk}^- \in \mathbb{R}^F$ are the dictionary filter, positive coefficient classifier, and negative coefficient classifier for the k-th component respectively. A key aspect of our formulation is that the optimal codes can be found very efficiently in closed-form—in a feed-forward manner (see Sec. B.2 for a detailed argument).

Asymmetric Shrinkage

To describe the coding processes, let us first define a generalized version of the shrinkage function commonly used in sparse coding. Our asymmetric shrinkage is parameterized by

upper and lower thresholds $-\beta^- \leq \beta^+$

$$\text{shrink}_{(\beta^+,\beta^-)}(v) = \begin{cases} v - \beta^+ & \text{if } v - \beta^+ > 0 \\ 0 & \text{otherwise} \\ v + \beta^- & \text{if } v + \beta^- < 0 \end{cases} \tag{3.9}$$



(a) $-\beta^- \leq 0 \leq \beta^+$ (b) $0 \leq -\beta^- \leq \beta^+$ (c) $-\beta^- \leq \beta^+ \leq 0$ (d) $\beta^- \leq 0 \leq -\beta^+$

Figure 3.2: Comparing the behavior of asymmetric shrinkage for different settings of β^+ and β^-. (a)-(c) satisfy the condition that $-\beta^- \leq \beta^+$ while (d) does not.

Fig. 3.2 shows a visualization of this function which generalizes the standard shrinkage proximal operator by allowing for the positive and negative thresholds. In particular, it corresponds to the proximal operator for a version of the ℓ_1-norm that penalizes the positive and negative components with different weights $|\mathbf{v}|_{asym} = \beta^+\|\mathbf{v}^+\|_1 + \beta^-\|\mathbf{v}^-\|_1$. The standard shrink operator corresponds to $\text{shrink}_{(\beta,-\beta)}(v)$ while the rectified linear unit common in CNNs is given by a limiting case $\text{shrink}_{(0,-\infty)}(v)$. We note that $-\beta^- \leq \beta^+$ is required for $\text{shrink}_{(\beta^+,\beta^-)}$ to be a proper function (see Fig. 3.2).

Feed-Forward Coding

We now describe how codes can be computed in a simple feed-forward pass. Let

$$\boldsymbol{\beta}^+_{yk} = \beta - \mathbf{w}^+_{yk}, \qquad \boldsymbol{\beta}^-_{yk} = \beta - \mathbf{w}^-_{yk} \tag{3.10}$$

be vectors of positive and negative biases whose entries are associated with a spatial location in the feature map k for class y. The optimal code $\mathbf{z}$ can be computed in three sequential steps:

1. Cross-correlate the data with the filterbank $\mathbf{d}_k \star \mathbf{x}$

2. Apply an asymmetric version of the standard shrinkage operator

$$\tilde{\mathbf{z}}_k = \text{shrink}_{(\boldsymbol{\beta}^+_{yk}, \boldsymbol{\beta}^-_{yk})}(\mathbf{d}_k \star \mathbf{x}) \tag{3.11}$$

   where, with abuse of notation, we allow the shrinkage function (Eq. 3.9) to apply entries in the vectors of threshold parameter pairs $\boldsymbol{\beta}^+_{yk}, \boldsymbol{\beta}^-_{yk}$ to the corresponding elements of the argument.

3. Project onto the feasible set of unit length codes

$$\mathbf{z}^* = \frac{\tilde{\mathbf{z}}}{\|\tilde{\mathbf{z}}\|_2}. \tag{3.12}$$

Relationship to CNNs:

We note that this formulation of coding has a close connection to single layer convolutional neural network (CNN). A typical CNN layer consists of convolution with a filterbank followed by a non-linear activation such as a rectified linear unit (ReLU). ReLUs can be viewed as another way of inducing sparsity, but rather than coring the values around zero like the shrink function, ReLU truncates negative values. On the other hand, the asymmetric shrink function can be viewed as the sum of two ReLUs applied to appropriately biased inputs:

$$\text{shrink}_{(\beta^+, \beta^-)}(x) = \text{ReLU}(x - \beta^+) - \text{ReLU}(-(x + \beta^-)),$$

SSC coding can thus be seen as a CNN in which the ReLU activation has been replaced with shrinkage followed by a global normalization.

3.3 Learning

We formulate supervised learning using the softmax log-loss that maximizes the energy for the true class label y_i while minimizing energy of incorrect labels $\bar{y}$.

$$\begin{aligned}
\underset{\mathbf{d},\mathbf{w}^+,\mathbf{w}^-,\beta\geq 0}{\arg\min}\ & \frac{\alpha}{2}(\|\mathbf{w}^+\|_2^2 + \|\mathbf{w}^-\|_2^2 + \|\mathbf{d}\|_2^2) \\
& + \frac{1}{N}\sum_{i=1}^{N}[-\max_{\|\mathbf{z}\|_2\leq 1} E(\mathbf{x}_i, y_i, \mathbf{z}) + \log\sum_{\bar{y}\in\mathcal{Y}} \max_{\|\bar{\mathbf{z}}\|_2\leq 1} e^{E(\mathbf{x}_i,\bar{y},\bar{\mathbf{z}})}], \qquad (3.13) \\
\text{s.t.}\quad & -(\beta - \mathbf{w}_{yk}^-) \leq (\beta - \mathbf{w}_{yk}^+) \quad \forall y, k
\end{aligned}$$

where α is the hyperparameter regularizing $\mathbf{w}_y^+$, $\mathbf{w}_y^-$, and $\mathbf{d}$. We constrain the relationship between β and the entries of $\mathbf{w}_y^+$ and $\mathbf{w}_y^-$ in order for the asymmetric shrinkage to be a proper function (see Sec. 3.2.3 and Sec. B.2 for details).

In classical sparse coding, it is typical to constrain the ℓ_2-norm of each dictionary filter to unit length. Our spherical coding objective behaves similarly. For any optimal code $\mathbf{z}^*$, there is a 1-dimensional subspace of parameters for which $\mathbf{z}^*$ is optimal given by scaling $\mathbf{d}$ inversely to $\mathbf{w}$, β. For simplicity of the implementation, we opt to regularize $\mathbf{d}$ to assure a unique solution. However, as [63] point out, it may be advantageous from the perspective of optimization to explicitly constrain the norm of the filterbank.

Note that unlike classical sparse coding, where β is a hyperparameter that is usually set using cross-validation, we treat it as a parameter of the model that is learned to maximize performance.

3.3.1 Optimization

In order to solve Eq. 3.13, we explicitly formulate our model as a directed-acyclic-graph (DAG) neural network with shared weights, where the forward-pass computes the sparse code vectors and the backward-pass updates the parameter weights. We optimize the objective using stochastic gradient descent (SGD).

As mentioned in Sec. 3.2.3 shrinkage function is asymmetric with parameters $\boldsymbol{\beta}^+_{yk}$ or $\boldsymbol{\beta}^-_{yk}$ as defined in Eq. 3.10. However, the inequality constraint on their relationship to keep the shrinkage function a proper function is difficult to enforce when optimizing with SGD. Instead, we introduce a central offset parameter and reduce the ordering constraint to pair of positivity constraints. Let

$$\hat{\mathbf{w}}^+_{yk} = \boldsymbol{\beta}^+_{yk} - b_k \qquad \hat{\mathbf{w}}^-_{yk} = \boldsymbol{\beta}^-_{yk} + b_k \tag{3.14}$$

be the modified linear "classifiers" relative to the central offset b_k. It is straightforward to see that if $\boldsymbol{\beta}^+_{yk}$ and $\boldsymbol{\beta}^-_{yk}$ that satisfy the constrain in Eq. 3.13, then adding the same value to both sides of the inequality will not change that. However, taking b_k to be a midpoint between them, then both $\boldsymbol{\beta}^+_{yk} - b_k$ and $\boldsymbol{\beta}^-_{yk} + b_k$ will be strictly non-negative.

Using this variable substitution, we rewrite the energy function (Eq. 3.1) as

$$E'(\mathbf{x}, y, \mathbf{z}) = \mathbf{x}^\intercal \big(\sum_{k=1}^{K} \mathbf{d}_k * \mathbf{z}_k \big) + \sum_{k=1}^{K} b_k \mathbf{1}^\intercal \mathbf{z}_k - \sum_{k=1}^{K} \hat{\mathbf{w}}^{+\intercal}_{yk} \mathbf{z}^+_k + \sum_{k=1}^{K} \hat{\mathbf{w}}^{-\intercal}_{yk} \mathbf{z}^-_k. \tag{3.15}$$

where $\mathbf{b}$ is constant offset for each code channel. The modified linear "classification" terms now take on a dual role of inducing sparsity and measuring the compatibility between $\mathbf{z}$ and y.

This yields a modified learning objective that can easily be solved with existing implemen-

tations for learning convolutional neural networks:

$$\begin{aligned}
&\arg\min_{\mathbf{d},\hat{\mathbf{w}}^+,\hat{\mathbf{w}}^-,\mathbf{b}} \frac{\alpha}{2}(\|\hat{\mathbf{w}}^+\|_2^2 + \|\hat{\mathbf{w}}^-\|_2^2 + \|\mathbf{d}\|_2^2) \\
&\qquad + \frac{1}{N}\sum_{i=1}^{N}[-\max_{\|\mathbf{z}\|_2\leq 1} E'(\mathbf{x}_i, y_i, \mathbf{z}) + \log\sum_{\bar{y}\in\mathcal{Y}} \max_{\|\bar{\mathbf{z}}\|_2\leq 1} e^{E'(\mathbf{x}_i,\bar{y},\bar{\mathbf{z}})}], \\
&\quad \text{s.t.} \;\; \hat{\mathbf{w}}_{yk}^+, \hat{\mathbf{w}}_{yk}^- \succeq 0 \quad \forall y, k
\end{aligned} \tag{3.16}$$

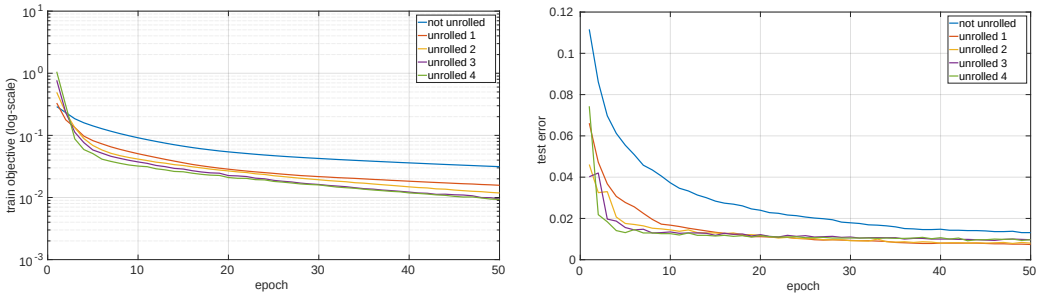
where $\hat{\mathbf{w}}^+$ and $\hat{\mathbf{w}}^-$ are the new sparsity inducing classifiers, and $\mathbf{b}$ are the arbitrary origin points. In particular, adding the K origin points allows us to enforce the constraint by simply projecting $\hat{\mathbf{w}}^+$ and $\hat{\mathbf{w}}^-$ onto the positive orthant during SGD.

Stacking Blocks

We also examine stacking multiple blocks of our energy function in order to build a hierarchical representation. As mentioned in Sec. 3.2.3, the optimal codes can be computed in a simple feed-forward pass—this applies to shallow versions of our model. When stacking multiple blocks of our energy-based model, solving for the optimal codes cannot be done in a feed-forward pass since the codes for different blocks are coupled (bilinearly) in the joint objective. Instead, we can proceed in an iterative manner, performing block-coordinate descent by repeatedly passing up and down the hierarchy updating the codes. In this section we investigate the trade-off between the number of passes used to find the optimal codes for the stacked model and classification performance.

For this purpose, we train multiple instances of a 2-block version of our energy-based model that differ in the number of iterations used when solving for the codes. For recurrent networks such as this, inference is commonly implemented by "unrolling" the network, where the parts of the network structure are repeated with parameters shared across these repeated parts to mimic an iterative algorithm that stops at a fixed number of iterations rather than at some

convergence criteria.



(a) Train Objective (b) Test Error

Figure 3.3: Comparing the effects of unrolling a 2-block version of our energy-based model. (Best viewed in color.)

In Fig. 3.3, we compare the performance between models that were unrolled zero to four times. We see that there is a difference in performance based on how many sweeps of the variables are made. In terms of the training objective, more unrolling produces models that have lower objective values with convergence after only a few passes. In terms of the testing error rate, however, we see that full code inference is not necessarily better, as unrolling once or twice has lower error rates than unrolling three or four times. The biggest difference was between not unrolling and unrolling once, where both the training objective and testing error rate go down. The testing error rate decreases from 0.0131 to 0.0074. While there is a clear benefit in terms of performance for unrolling at least once, there is also a trade-off between performance and computational resource, especially for deeper models.

3.4 Experiments

We evaluate the benefits of combining top-down and bottom-up information to produce class-specific features on the CIFAR-10 [37] dataset using a deep version of our EB-SSC. All experiments were performed using the MatConvNet [64] framework with the ADAM optimizer [30]. The data was preprocessed and augmented following the procedure in [18].

Base Network		
block	kernel, stride, padding	activation
conv1	$3 \times 3 \times 3 \times 96, 1, 1$	ReLU/CReLU
conv2	$3 \times 3 \times 96/192 \times 96, 1, 1$	ReLU/CReLU
pool1	$3 \times 3, 2, 1$	max
conv3	$3 \times 3 \times 96/192 \times 192, 1, 1$	ReLU/CReLU
conv4	$3 \times 3 \times 192/384 \times 192, 1, 1$	ReLU/CReLU
conv5	$3 \times 3 \times 192/384 \times 192, 1, 1$	ReLU/CReLU
pool2	$3 \times 3, 2, 1$	max
conv6	$3 \times 3 \times 192/384 \times 192, 1, 1$	ReLU/CReLU
conv7	$1 \times 1 \times 192/384 \times 192, 1, 1$	ReLU/CReLU

Table 3.1: Underlying block architecture common across all models we evaluated. SSC networks add an extra normalization layer after the non-linearity. And EB-SSC networks insert class-specific bias layers between the convolution layer and the non-linearity. Concatenated ReLU (CReLU) splits positive and negative activations into two separate channels rather than discarding the negative component as in the standard ReLU.

Specifically, the data was made zero mean and whitened, augmented with horizontal flips (with a 0.5 probability) and random cropping. No weight decay was used, but we used a dropout rate of 0.3 before every convolution layer except for the first. For these experiments we consider a single forward pass (no unrolling).

3.4.1 Classification

We compare our proposed EB-SSC model to that of [61], which uses rectified linear units (ReLU) as its non-linearity. This model can be viewed as a basic feed-forward version of our proposed model which we take as a baseline. We also consider variants of the baseline model that utilize a subset of architectural features of our proposed model (*e.g.*, concatenated rectified linear units (CReLU) and spherical normalization (SN)) to understand how subtle design changes of the network architecture affects performance.

We describe the model architecture in terms of the feature extractor and classifier. Table 3.1 shows the overall network architecture of feature extractors, which consist of seven

convolution blocks and two pooling layers. We test two possible classifiers: a simple linear classifier (LC) and our energy-based classifier (EBC), and use softmax-loss for all models. For linear classifiers, a numerical subscript indicates which of the seven conv blocks of the feature extractor is used for classification (*e.g.*, LC_7 indicates the activations out of the last conv block is fed into the linear classifier). For energy-based classifiers, a numerical subscript indicates which conv blocks of the feature extractor are replace with a energy-based classifier (*e.g.*, EBC_{6-7} indicates the activations out of conv5 is fed into the energy-based classifier and the energy-based classifier has a similar architecture to the conv blocks it replaces). The notation differs because for energy-based classifiers, the optimal activations are a function of the hypothesized class label, whereas for linear classifiers, they are not.

Model	Train Err. (%)	Test Err. (%)	# params
ReLU+LC_7	1.20	11.40	1.3M
CReLU+LC_7	2.09	10.17	2.6M
CReLU(SN)+LC_7	0.99	9.74	2.6M
SSC+LC_7	0.99	9.77	2.6M
SSC+EBC_{6-7}	0.21	9.23	3.2M

Table 3.2: Comparison of the baseline ReLU+LC_7 model, its derivative models, and our proposed model on CIFAR-10.

The results shown in Table 3.2 compare our proposed model to the baselines ReLU+LC_7 [61] and CReLU+LC_7 [58], and to intermediate variants. The baseline models all perform very similarly with some small reductions in error rates over the baseline CReLU+LC_7. However, CReLU+LC_7 reduces the error rate over ReLU+LC_7 by more than one percent (from 11.40% to 10.17%), which confirms the claims by [58] and demonstrates the benefits of splitting positive and negative activations. Likewise, we see further decrease in the error rate (to 9.74%) from using spherical normalization. Though normalizing the activations doesn't add any capacity to the model, this improved performance is likely because scale-invariant activations makes training easier. On the other hand, further sparsifying the activations yielded no benefit. We tested values $\beta = \{0.001, 0.01\}$ and found 0.001 to perform better. Replacing the linear classifier with our energy-based classifier further decreases the error rate

by another half percent (to 9.23%).

3.4.2 Decoding Class-Specific Codes

A unique aspect of our model is that it is generative in the sense that each layer is explicitly trying to encode the activation pattern in the prior layer. Similar to the work on deconvolutional networks built on least-squares sparse coding [74], we can synthesize input images from activations in our spherical coding network by performing repeated deconvolutions (transposed convolutions) back through the network. Since our model is energy based, we can further examine how the top-down information of a hypothesized class effects the intermediate activations.

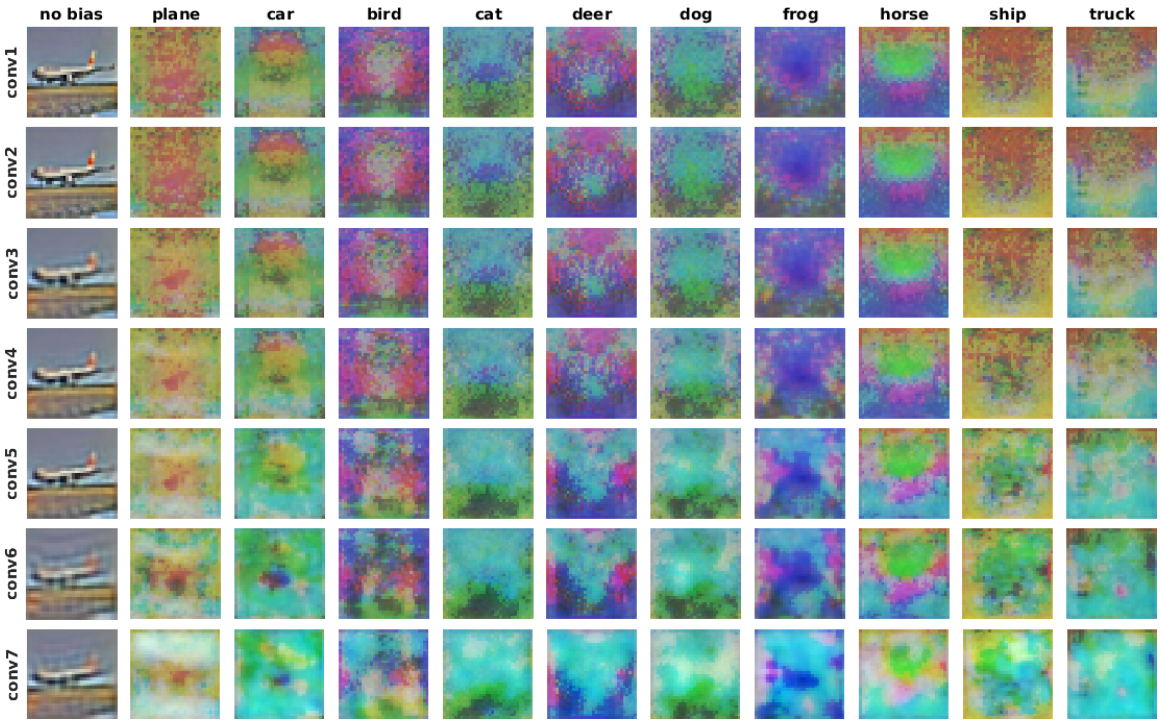
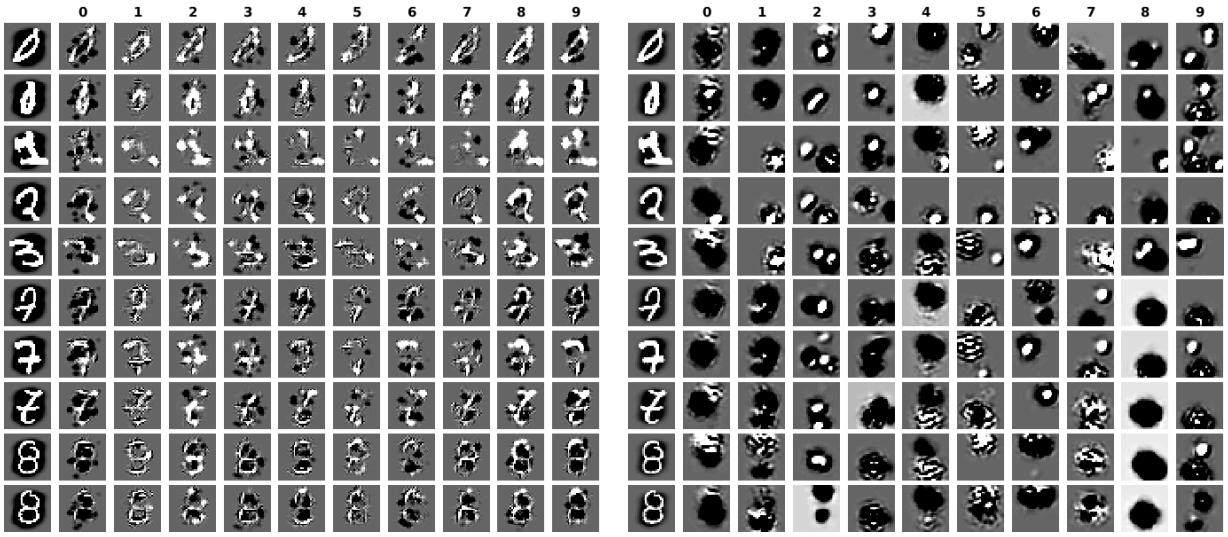


Figure 3.4: The reconstruction of an airplane image from different levels of the network (rows) across different hypothesized class labels (columns). The first column is pure reconstruction, *i.e.*, unbiased by a hypothesized class label, the remaining columns show reconstructions of the learned class bias at each layer for one of ten possible CIFAR-10 class labels. (Best viewed in color.)

The first column in Fig. 3.4 visualizes reconstructions of a given input image based on activations from different layers of the model by convolution transpose. In this case we put in zeros for class biases (*i.e.*, no top-down) and are able to recover high fidelity reconstructions of the input. In the remaining columns, we use the same deconvolution pass to construct input space representations of the learned classifier biases. At low levels of the feature hierarchy, these biases are spatially smooth since the receptive fields are small and there is little spatial invariance capture in the activations. At higher levels these class-conditional bias fields become more tightly localized.

Finally, in Fig. 3.5 we shows decodings from the conv2 and conv5 layer of the EB-SSC model for a given input under different class hypotheses. Here we subtract out the contribution of the top-down bias term in order to isolate the effect of the class conditioning on the encoding of input features. As visible in the figure, the modulation of the activations focused around particular regions of the image and the differences across class hypotheses becomes more pronounced at higher layers of the network.



(a) conv2 (b) conv5

Figure 3.5: Visualizing the reconstruction of different input images (rows) for each of 10 different class hypotheses (cols) from the 2nd and 5th block activations for a model trained on MNIST digit classification.

3.5 Conclusion

We presented an energy-based sparse coding method that efficiently combines cosine similarity, convolutional sparse coding, and linear classification. Our model shows a clear mathematical connection between the activation functions used in CNNs to introduce sparsity and our cosine similarity convolutional sparse coding formulation. Our proposed model outperforms the baseline model and we show which attributes of our model contributes most to the increase in performance. We also demonstrate that our proposed model provides an interesting framework to probe the effects of class-specific coding.

Chapter 4

Cross-Domain Image Matching with Deep Features

In Chapters 2 & 3 we focused on learning features for specific data and tasks. We now shift our focus to learning methods that utilize the information already captured in existing features. In this chapter, we show features extracted from pretrained convolutional neural networks can be effective for other tasks when the task-specific function (even with no learned parameters) is sensibly chosen. The features are surprisingly effective considering these models were trained for tasks and on data completely different from our own.

4.1 Introduction

We investigate the problem of automatically determining what type (brand/model/size) of shoe left an impression found at a crime scene. In the forensic footwear examination literature [2], this fine-grained category-level recognition problem is known as determining the *class characteristics* of a tread impression. This is distinct from the instance-level recognition

problem of matching *acquired characteristics* such as cuts or scratches which can provide stronger evidence that a specific shoe left a specific mark.

Analysis of shoe tread impressions is made difficult by the variability in types of crime scene evidence (ranging from traces of dust or oil on hard surfaces to impressions made in soil) and the lack of comprehensive datasets of shoe outsole tread patterns (see Fig. 4.1). Solving this problem requires developing models that can handle *cross-domain* matching of tread features between photos of clean test impressions (or images of shoe outsoles) and photos of crime scene evidence. We face the additional challenge that we would like to use extracted image features for matching a given crime scene impression to a large, open-ended database of exemplar tread patterns.

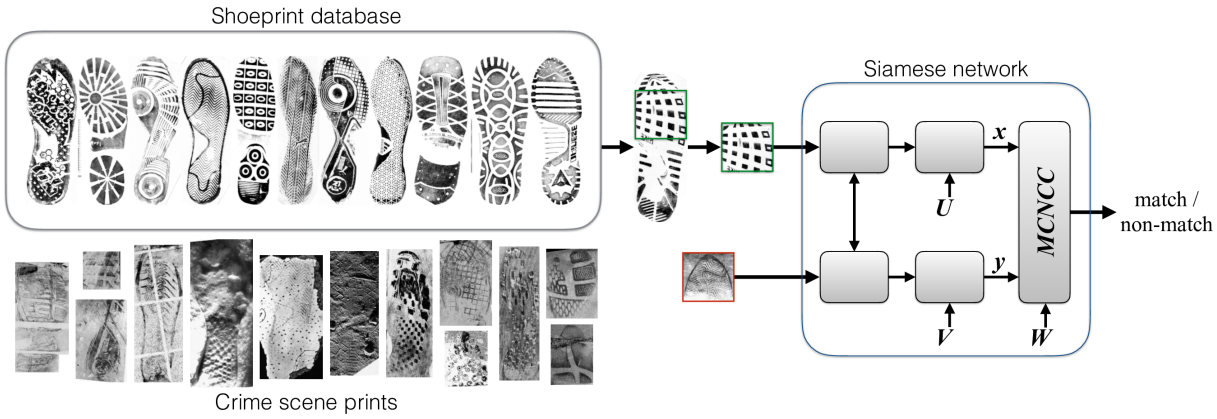


Figure 4.1: We would like to match crime scene prints to a database of test impressions despite significant cross-domain differences in appearance. We utilize a Siamese network to perform matching using a multi-channel normalized cross correlation. We find that per-exemplar, per-channel normalization of CNN feature maps significantly improves matching performance. Here U and V are the linear projection parameters for laboratory test impression and crime scene photo domains respectively. W is the per-channel importance weights. And x and y are the projected features of each domain used for matching.

Cross-domain image matching arises in a variety of other application domains beyond our specific scenario of forensic shoeprint matching. For example, matching aerial photos to GIS map data for location discovery [57, 7, 10], image retrieval from hand drawn sketches and paintings [5, 60], and matching images to 3D models [56]. As with shoeprint matching,

many of these applications often lack large datasets of ground-truth examples of cross-domain matches. This lack of training data makes it difficult to learn cross-domain matching metrics directly from raw pixel data. Instead traditional approaches have focused on designing feature extractors for each domain which yield domain invariant descriptions (e.g., locations of edges) which can then be directly compared.

Deep convolutional neural net (CNN) features hierarchies have proven incredibly effective at a wide range of recognition tasks. Generic feature extractors trained for general-purpose image categorization often perform surprising well for novel categorization tasks without performing any fine-tuning beyond training a linear classifier [59]. This is often explained by appealing to the notion that these learned representations extract image features with invariances that are, in some sense, generic. We might hope that these same invariances would prove useful in our setting (*e.g.*, encoding the shape of a tread element in a way that is insensitive to shading, contrast reversals, etc.). However, our problem differs in that we need to formulate a cross-domain similarity metric rather than simply training a k-way classifier.

Building on our previous work [33], we tackle this problem using similarity measures that are derived from normalized cross-correlation (NCC), a classic approach for matching gray-scale templates. For CNN feature maps, it is necessary to extend this to handle multiple channels. Our contribution is to propose a multi-channel variant of NCC which performs normalization on a per-channel basis (rather than, e.g., per-feature volume). We find this performs substantially better than related similarity measures such as the widely used cosine distance. We explain this finding in terms of the statistics of CNN feature maps. Finally, we use this multi-channel NCC as a building block for a Siamese network model which can be trained end-to-end to optimize matching performance.

4.2 Related Work

Shoeprint recognition The widespread success of automatic fingerprint identification systems (AFIS) [41] has inspired many attempts to similarly automate shoeprint recognition. Much initial work in this area focused on developing feature sets that are rotation and translation invariant. Examples include, phase only correlation [20], edge histogram DFT magnitudes [75], power spectral densities [9, 8], and the Fourier-Mellin transform [20]. Some other approaches pre-align the query and database image using the Radon transform [49] while still others sidestep global alignment entirely by computing only relative features between keypoints pairs [62, 50]. Finally, alignment can be implicitly computed by matching rotationally invariant keypoint descriptors between the query and database images [50, 65]. The recent study of Richetelli *et al.* [54] carries out a comprehensive evaluation of many of these approaches in a variety of scenarios using a carefully constructed dataset of crime scene-like impressions. In contrast to these previous works, we handle global invariance by explicitly matching templates using dense search over translations and rotations.

One-shot learning While we must match our crime scene evidence against a large database of candidate shoes, our database contains very few examples per-class. As such, we must learn to recognize each shoe category with as little as one training example. This can be framed as a one-shot learning problem [42]. Prior work has explored one-shot object recognition with only a single training example, or "exemplar" [44]. Specifically in the domain of shoeprints, Kortylewski *et al.* [36] fit a compositional active basis model to an exemplar which could then be evaluated against other images. Alternatively, standardized or whitened off-the-shelf HOG features have proven very effective for exemplar recognition [21]. Our approach is similar in that we examine the performance of one-shot recognition using generic deep features which have proven surprisingly robust for a huge range of recognition tasks [59].

Similarity metric learning While off-the-shelf deep features work well [59], they can be often be *fine tuned* to improve performance on specific tasks. In particular, for a paired comparison tasks, so-called "Siamese" architectures integrate feature extraction and comparison in a single differentiable model that can be optimized end-to-end. Past work has demonstrated that Siamese networks learn good features for person re-identification, face recognition, and stereo matching [73, 48, 67]; deep pseudo-Siamese architectures can even learn to embed two dissimilar domains into a common co-domain [72]. For shoe class recognition, we similarly learn to embed two types of images: (1) crime scene photos and (2) laboratory test impressions.

4.3 Multi-variate Cross Correlation

In order to compare two corresponding image patches, we extend the approach of normalized cross-correlation (often used for matching gray-scale images) to work with multi-channel CNN features. Interestingly, there is not an immediately obvious extension of NCC to multiple channels, as evidenced by multiple approaches proposed in the literature [14, 46, 16, 51]. To motivate our approach, we appeal to a statistical perspective.

Normalized correlation Let x, y be two scalar random variables. A standard measure of correlation between two variables is given by their *Pearson's correlation coefficient* [46]:

$$\rho(x, y) = E[\tilde{x}\tilde{y}] = \frac{\sigma_{xy}}{\sqrt{\sigma_{xx}}\sqrt{\sigma_{yy}}}, \tag{4.1}$$

where

$$\tilde{x} = \frac{x - \mu_x}{\sqrt{\sigma_{xx}}}$$

is the *standardized* version of x (similarly for y) and

$$\mu_x = E[x], \qquad \sigma_{xx} = E[(x-\mu_x)^2], \qquad \sigma_{xy} = E[(x-\mu_x)(y-\mu_y)].$$

Intuitively, the above corresponds to the correlation between two transformed random variables that are "whitened" to have zero-mean and unit variance. The normalization ensures that correlation coefficient will lie between -1 and $+1$.

Normalized cross-correlation Let us model pixels x from an image patch X as corrupted by some i.i.d. noise process and similarly pixels another patch Y (of identical size) as y. The *sample* estimate of the Pearson's coefficient for variables x, y is equivalent to the normalized cross-correlation (NCC) between patches X, Y:

$$\mathrm{NCC}(X,Y) = \frac{1}{|P|} \sum_{i \in P} \frac{(x[i]-\mu_x)}{\sqrt{\sigma_{xx}}} \frac{(y[i]-\mu_y)}{\sqrt{\sigma_{yy}}}, \tag{4.2}$$

where P refers to the set of pixel positions in a patch and means and standard deviations are replaced by their sample estimates.

From the perspective of detection theory, normalization is motivated by the need to compare correlation coefficients across different pairs of samples with non-stationary statistics (*e.g.*, determining which patches $\{Y^1, Y^2, \ldots\}$ are the same as a given template patch X where statistics vary from one Y to the next). Estimating first and second-order statistics per-patch provides a convenient way to handle sources of "noise" that are approximately i.i.d. conditioned on the choice of patch P but not independent of patch location.

Multivariate extension Let us extend the above formulation for random *vectors* $\mathbf{x}, \mathbf{y} \in R^N$ where N corresponds to the multiple channels of values at each pixel (*e.g.*, $N = 3$ for a RGB image). The scalar correlation is now replaced by a $N \times N$ correlation *matrix*. To

produce a final score capturing the overall correlation, we propose to use the *trace* of this matrix, which is equivalent to the sum of its eigenvalues. As before, we add invariance by computing correlations on transformed variables $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}$ that are "whitened" to have a zero-mean and identity covariance matrix:

$$\rho_{multi}(\mathbf{x}, \mathbf{y}) = \frac{1}{N} Tr(E[\tilde{\mathbf{x}}\tilde{\mathbf{y}}^T]) = \frac{1}{N} Tr(\Sigma_{\mathbf{xx}}^{-\frac{1}{2}} \Sigma_{\mathbf{xy}} \Sigma_{\mathbf{yy}}^{-\frac{1}{2}})$$

where:

$$\tilde{\mathbf{x}} = \Sigma_{\mathbf{xx}}^{-\frac{1}{2}}(\mathbf{x} - \mu_{\mathbf{x}}), \quad \Sigma_{\mathbf{xx}} = E[(\mathbf{x} - \mu_{\mathbf{x}})(\mathbf{x} - \mu_{\mathbf{x}})^T], \quad \Sigma_{\mathbf{xy}} = E[(\mathbf{x} - \mu_{\mathbf{x}})(\mathbf{y} - \mu_{\mathbf{y}})^T]. \tag{4.3}$$

The above multivariate generalization of the Pearson's coefficient is arguably rather natural, and indeed, is similar to previous formulations that also make use of a trace operator on a correlation matrix [46, 51]. However, one crucial distinction from such past work is that our generalization (4.3) reduces to (4.1) for $N = 1$. In particular, [46, 51] propose multivariate extensions that are restricted to return a nonnegative coefficient. It is straightforward to show that our multivariate coefficient will lie between -1 and $+1$.

Decorrelated channel statistics The above formulation can be computationally cumbersome for large N, since it requires obtaining sample estimates of matrices of size N^2. Suppose we make the strong assumption that all N channels are *uncorrelated* with each other. This greatly simplifies the above expression, since the covariance matrices are then diagonal matrices:

$$\Sigma_{\mathbf{xy}} = \text{diag}(\{\sigma_{x_c y_c}\}), \qquad \Sigma_{\mathbf{xx}} = \text{diag}(\{\sigma_{x_c x_c}\}), \qquad \Sigma_{\mathbf{yy}} = \text{diag}(\{\sigma_{y_c y_c}\}).$$

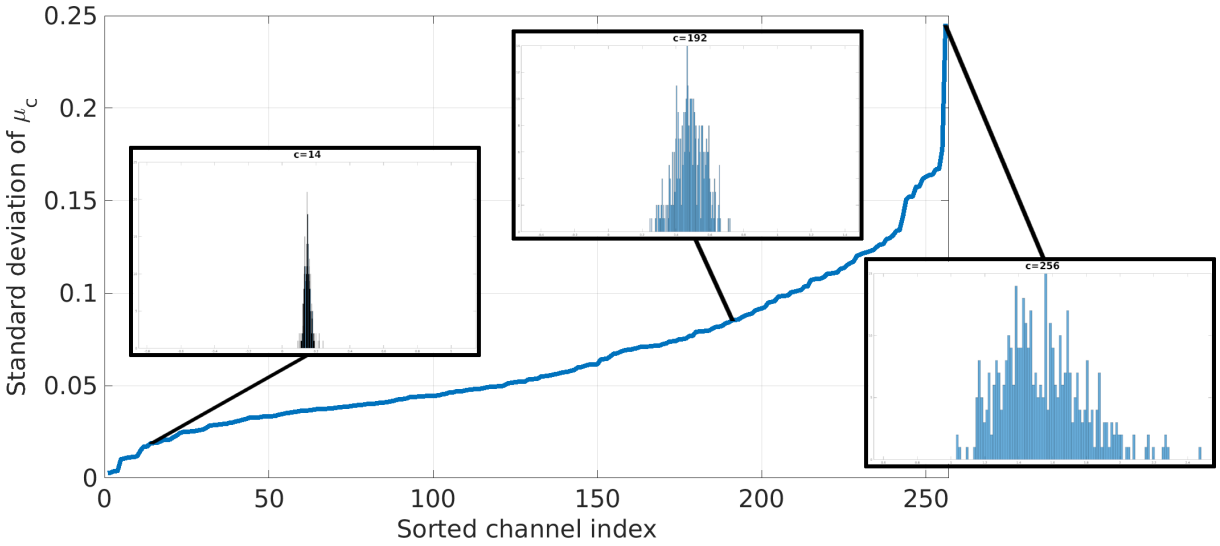


Figure 4.2: **Distribution of patch channel means:** For each query image (patch) we match against the database, our proposed MCNCC similarity measure normalizes ResNet-50 'res2x' feature channels by their individual mean and standard deviation. For uniformly sampled patches, we denote the normalizing mean for channel c using the random variable μ_c. For each channel, we plot the standard deviation of μ_c above with channels sorted by increasing standard deviation. When the mean response for a channel varies little from one patch to the next (small std, left), we can expect that a global, per-dataset transformation (*e.g.*, PCA or CCA whitening) is sufficient to normalize the channel response. However, for channels where individual patches in the dataset have very different channel means (large std, right), normalizing by the local (per-patch) statistics provides additional invariance.

Plugging this assumption into (4.3) yields the simplified expression for multivariate correlation

$$\rho_{multi}(\mathbf{x}, \mathbf{y}) = \frac{1}{N} \sum_{c=1}^{N} \frac{\sigma_{x_c y_c}}{\sqrt{\sigma_{x_c x_c}} \sqrt{\sigma_{y_c y_c}}} \tag{4.4}$$

where the diagonal multivariate statistic is simply the average of N per-channel correlation coefficients. It is easy to see that this sum must lie between -1 and $+1$.

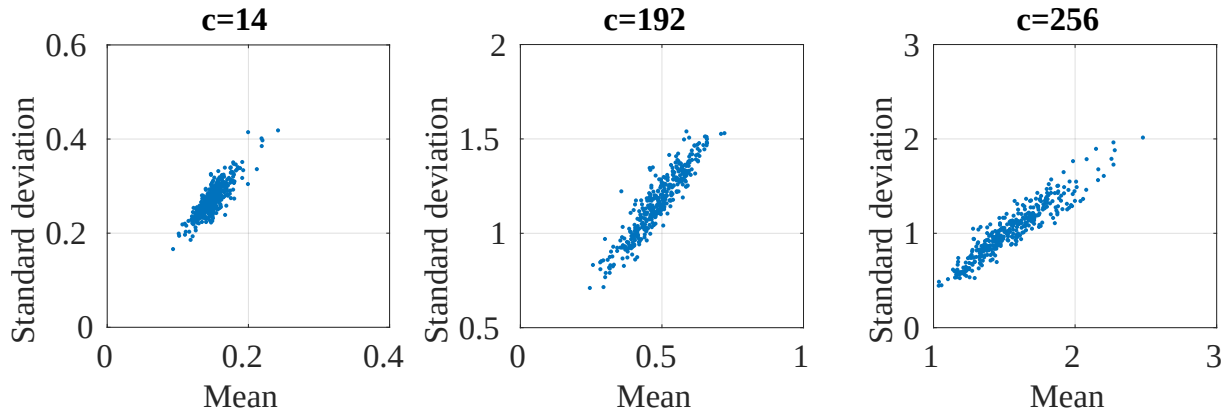


Figure 4.3: **Normalizing channel statistics:** As shown in the histograms of Fig. 4.2, for some feature channels, patches have wildly different means and standard deviations. For channel 14 (left), the statistics (and hence normalization) are similar from one patch to the next while for channel 256 (right), means and standard deviations vary substantially across patches. CNN channel activations are positive so means and standard deviations are strongly correlated.

Multi-channel NCC The sample estimate of (4.4) yields a multi-channel extension of NCC which is adapted to the patch:

$$\text{MCNCC}(X, Y) = \frac{1}{N|P|} \sum_{c=1}^{N} \sum_{i \in P} \frac{(x_c[i] - \mu_{x_c})}{\sqrt{\sigma_{x_c x_c}}} \frac{(y_c[i] - \mu_{y_c})}{\sqrt{\sigma_{y_c y_c}}}$$

The above multi-channel extension is similar to the final formulation in [14], but is derived from a statistical assumption on the channel correlation.

Cross-domain covariates and whitening Assuming a diagonal covariance makes strong assumptions about cross-channel correlations. When strong cross-correlations exist, an alternative approach to reducing computational complexity is to assume that cross-channel correlations lie within a K dimensional subspace, where $K \leq N$. We can learn a projection matrix for reducing the dimensionality of features from both patch X and Y which

decorrelates and scales the channels to have unit variance:

$$\hat{\mathbf{x}} = U(\mathbf{x} - \mu_x), \quad U \in R^{K \times N}, \quad E[\hat{\mathbf{x}}\hat{\mathbf{x}}^T] = I$$
$$\hat{\mathbf{y}} = V(\mathbf{y} - \mu_y), \quad V \in R^{K \times N}, \quad E[\hat{\mathbf{y}}\hat{\mathbf{y}}^T] = I.$$

In general, the projection matrix could be different for different domains (in our case, crime scene versus test prints). One strategy for learning the projection matrices is applying principle component analysis (PCA) on samples from each domain separately. Alternatively, when paired training examples are available, one could use canonical correlation analysis (CCA) [45], which jointly learn the projections that maximize correlation across domains. An added benefit of using *orthogonalizing* transformations such as PCA/CCA is that transformed data satisfies the diagonal assumptions (globally) allowing us to estimate patch multivariate correlations in this projected space with diagonalized covariance matrices of size $K \times K$.

Global versus local whitening There are two distinct aspects to whitening (or normalizing) variables in our problem setup to be determined: (1) assumptions on the structure of the sample mean and covariance matrix, and (2) the data over which the sample mean and covariance are estimated. In choosing the structure, one could enforce an unrestricted covariance matrix, a low-rank covariance matrix (e.g., PCA), or a diagonal covariance matrix (e.g., estimating scalar means and variances). In choosing the data, one could estimate these parameters over individual patches (local whitening) or over the entire dataset (global whitening). In Section 4.5, we empirically explore various combinations of these design choices which are computationally feasible (e.g., estimating a full-rank covariance matrix locally for each patch would be too expensive). We find a good tradeoff to be global whitening (to decorrelate features globally), followed by local whitening with a diagonal covariance assumption (e.g., MCNCC).

To understand the value of global and per-patch normalization, we examine the statistics of CNN feature channels across samples of our dataset. Fig. 4.2 and Fig. 4.3 illustrate how the per-channel normalizing statistics (μ_c, σ_c) vary across patches and across channels. Notably, for some channels, the normalizing statistics change substantially from patch to patch. This makes the results of performing local, per-patch normalization significantly different from global, per-dataset normalization.

One common effect of both global and local whitening is to prevent feature channels that tend to have large means and variances from dominating the correlation score. However, by the same merit this can have the undesirable effect of amplifying the influence of low-variance channels which may not be discriminative for matching. In the next section we generalize both PCA and CCA using a learning framework which can learn channel decorrelation and per-channel importance weighting by optimizing a discriminative performance objective.

4.4 Learning Correlation Similarity Measures

In order to allow for additional flexibility of weighting the relevance of each channel we consider a channel-weighted variant of MCNCC parameterized by vector W:

$$\text{MCNCC}_W(X, Y) = \sum_{c=1}^{N} W_c \left[\frac{1}{|P|} \sum_{i \in P} \frac{(x_c[i] - \mu_{x_c})}{\sqrt{\sigma_{x_c x_c}}} \frac{(y_c[i] - \mu_{y_c})}{\sqrt{\sigma_{y_c y_c}}} \right]. \tag{4.5}$$

This per-channel weighting can undo the effect of scaling by the standard deviation in order to re-weight channels by their informativeness. Furthermore, since the features x, y are themselves produced by a CNN model, we can consider the parameters of that model as additional candidates for optimization. In this view, PCA/CCA can be seen as adding an extra linear network layer prior to the correlation calculation. The parameters of such a layer can be initialized using PCA/CCA and then discriminatively tuned. The resulting

"Siamese" architecture is illustrated in Fig. 4.1.

Siamese loss: To train the model, we minimize a hinge-loss:

$$\begin{aligned}\underset{W,U,V,b}{\arg\min} \quad & \frac{\alpha}{2}\|W\|_2^2 + \frac{\beta}{2}\left(\|U\|_F^2 + \|V\|_F^2\right) \\ & + \sum_{s,t} \max\left(0, 1 - z_{s,t}\,\mathrm{MCNCC}_W(\phi_U(X^s), \phi_V(Y^t)) + b\right)\end{aligned} \tag{4.6}$$

where we have made explicit the function ϕ which computes the deep features of two shoeprints X^s and Y^t, with W, U, and V representing the parameters for the per-channel importance weighting and the linear projections for the two domains respectively. b is the bias and $z_{s,t}$ is a binary same-source label (*i.e.*, $+1$ when X^s and Y^t come from the same source and -1 otherwise). Finally, α is the regularization hyperparameter for W and β is the same for U and V.

We implement ϕ using a deep architecture, which is trainable using standard backpropagation. Each channel contributes a term to the MCNCC which itself is just a single channel (NCC) term. The operation is symmetric in X and Y, and the gradient can be computed efficiently by reusing the NCC computation from the forward pass:

$$\frac{d\,\mathrm{NCC}(x_c, y_c)}{d\,x_c[j]} = \frac{1}{|P|\sqrt{\sigma_{x_c x_c}}}(\tilde{y}_c[j] + \tilde{x}_c[j]\,\mathrm{NCC}(x_c, y_c)). \tag{4.7}$$

Derivation of NCC gradient: To derive the NCC gradient, we first expand it as a sum over individual pixels indexed by i and consider the total derivative with respect to input feature $x[j]$:

$$\frac{d\,\mathrm{NCC}(x, y)}{d\,x[j]} = \frac{1}{|P|}\sum_{i \in P} \tilde{y}[i]\left(\frac{\partial \tilde{x}[i]}{\partial x[j]} + \frac{\partial \tilde{x}[i]}{\partial \mu_x}\frac{\partial \mu_x}{\partial x[j]} + \frac{\partial \tilde{x}[i]}{\partial \sigma_{xx}}\frac{\partial \sigma_{xx}}{\partial x[j]}\right) \tag{4.8}$$

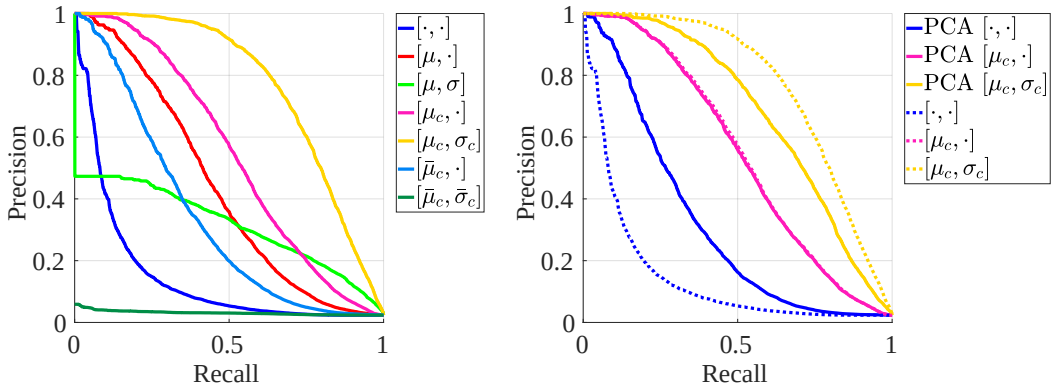


Figure 4.4: Comparing MCNCC to baselines for image retrieval within the same domain. The methods are denoted by two operations in square brackets: centering and normalization, respectively. μ and σ denote computing the statistics across all channels, μ_c and σ_c denote computing per-channel statistics, and $\cdot$ denotes the absence of the operation (*e.g.*, MCNCC is denoted as $[\mu_c, \sigma_c]$, whereas cross-correlation is denoted as $[\cdot, \cdot]$. Finally, $\bar{\mu}_c$ and $\bar{\sigma}_c$ denote computing the average per-channel statistics across the dataset. The left panel shows the performance on the raw features, whereas the right panel compares globally whitened features using PCA (solid lines) against their corresponding raw features (dotted lines). (Best viewed in color.)

, where we have have dropped the channel subscript for clarity. The partial derivative $\frac{\partial \tilde{x}[i]}{\partial x[j]} = \frac{1}{\sqrt{\sigma_{xx}}}$, if and only if $i = j$ and is zero otherwise. The remaining partials derive as follows:

$$\frac{\partial \tilde{x}[i]}{\partial \mu_x} = -\frac{1}{\sqrt{\sigma_{xx}}} \qquad \frac{\partial \mu_x}{\partial x[j]} = \frac{1}{|P|}$$

$$\frac{\partial \tilde{x}[i]}{\partial \sigma_{xx}} = \frac{1}{2\sigma_{xx}^{3/2}}(x[i] - \mu_x) \qquad \frac{\partial \sigma_{xx}}{\partial x[j]} = \frac{2\,(x[j] - \mu_x)}{|P|}.$$

Substituting them into Eq. 4.8, we arrive at a final expression:

$$\begin{aligned}
\frac{d\,\mathrm{NCC}(x,y)}{d\,x[j]} &= \frac{\tilde{y}[j]}{|P|\sqrt{\sigma_{xx}}} + \frac{1}{|P|}\sum_{i\in P}\tilde{y}[i]\left(\frac{-1}{|P|\sqrt{\sigma_{xx}}} + \frac{2\,(x[i]-\mu_x)\,(x[j]-\mu_x)}{2|P|\sigma_{xx}^{3/2}}\right) \\
&= \frac{1}{|P|\sqrt{\sigma_{xx}}}\left(\tilde{y}[j] + \frac{1}{|P|}\sum_{i\in P}\tilde{y}[i]\left(-1 + \frac{(x[i]-\mu_x)\,(x[j]-\mu_x)}{\sigma_{xx}}\right)\right) \\
&= \frac{1}{|P|\sqrt{\sigma_{xx}}}\left(\tilde{y}[j] - \frac{1}{|P|}\sum_{i\in P}\tilde{y}[i] + \frac{1}{|P|}\sum_{i\in P}\tilde{y}[i]\tilde{x}[i]\tilde{x}[j]\right) \\
&= \frac{1}{|P|\sqrt{\sigma_{xx}}}(\tilde{y}[j] + \tilde{x}[j]\,\mathrm{NCC}(x,y)), \qquad (4.9)
\end{aligned}$$

where we have made use of the fact that $\tilde{y}$ is zero-mean.

4.5 Diagnostic Experiments

To understand the effects of feature channel normalization on retrieval performance, we compare the proposed MCNCC measure to two baseline approaches: simple unnormalized cross-correlation and cross-correlation normalized by a single μ and σ estimated over the whole 3D feature volume. We note that the latter is closely related to the "cosine similarity" which is popular in many retrieval applications (cosine similarity scales by σ but does not subtract μ). We also consider variants which only perform partial standardization and/or whitening of the input features.

Partial print matching: We evaluate these methods in a setup that mimics the occurrence of partial occlusions in shoeprint matching, but focus on a single modality of test impressions. We extract 512 query patches (random selected 97×97 pixel sub-windows) from test impressions that have two or more matching tread patterns in the database. The task is then to retrieve from the database the set of relevant prints. As the query patches are smaller than the test impressions, we search over spatial translations (with a stride of 1),

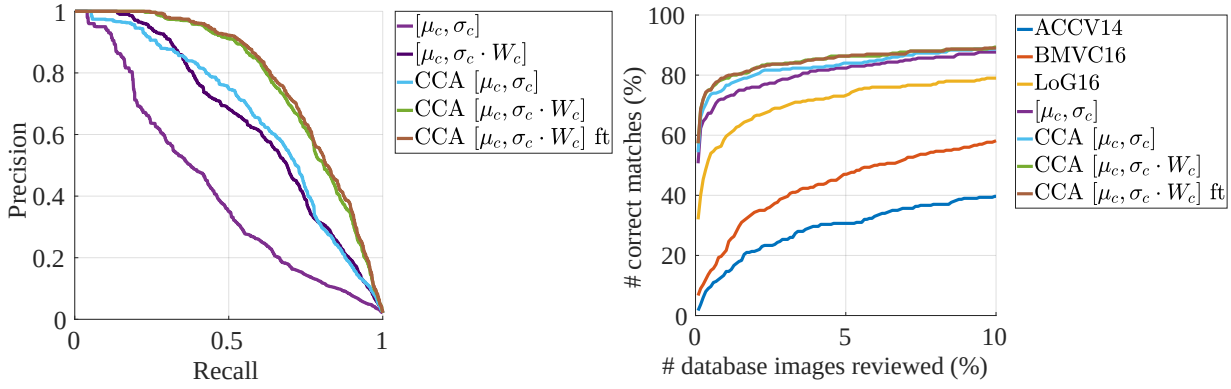


Figure 4.5: Comparing MCNCC with uniform weights (denoted as $[\mu_c, \sigma_c]$), learned per-channel weights (denoted as $[\mu_c, \sigma_c \cdot W_c]$), learned linear projections (denoted as CCA $[\mu_c, \sigma_c]$), piece-wise learned projection and per-channel weights (denoted as CCA $[\mu_c, \sigma_c \cdot W_c]$), and jointly learned projection and per-channel weights (denoted as CCA $[\mu_c, \sigma_c \cdot W_c]$ ft) for retrieving relevant shoeprint test impressions for crime scene prints. The left panel shows our five methods on the Israeli dataset. The right panel compares variants of our proposed system against the current state-of-the-art, as published in: ACCV14 [35], BMVC16 [36] and LoG16 [34] using cumulative match characteristic (CMC).

using the maximizing correlation value to score the match to the test impression. We do not need to search over rotations as all test impressions were aligned to a canonical orientation. When querying the database, the original shoeprint the query was extracted from is removed (*i.e.*, the results do not include the self-match).

We carry out these experiments using a dataset that contains 387 test impression of shoes and 137 crime scene prints collected by the Israel National Police [71]. As this dataset is not publicly available, we used this dataset primarily for the diagnostic analysis and for training and validating learned models. In these diagnostic experiments, except where noted otherwise, we use the 256-channel 'res2bx' activations from a pre-trained ResNet-50 model[1]. We evaluated feature maps at other locations along the network, but found those to performed the best.

[1]Pretrained model was obtained from `http://www.vlfeat.org/matconvnet/models/imagenet-resnet-50-dag.mat`

Global versus local normalization: Fig. 4.4 shows retrieval performance in terms of the tradeoff of precision and recall at different match thresholds. In the legend we denote different schemes in square brackets, where the first term indicates the centering operation and the second term indicates the normalization operation. A $\cdot$ indicates the absence of the operation. μ and σ indicate that standardization was performed using local (*i.e.*, *per-exemplar*) statistics of features over the entire (3D) feature map. μ_c and σ_c indicate local per-channel centering and normalization. $\bar{\mu}_c$ and $\bar{\sigma}_c$ indicate global per-channel centering and normalization (*i.e.*, statistics are estimated over the *whole dataset*). Therefore, simple unnormalized cross-correlation is indicated as $[\cdot, \cdot]$, cosine distance is indicated as $[\mu, \sigma]$, and our proposed MCNCC measure is indicated as $[\mu_c, \sigma_c]$.

We can clearly see from the left panel of Fig. 4.4 that using per-channel statistics estimated independently for each comparison gives substantial gains over the baseline methods. Centering using 3D (across-channel) statistics is better than either centering using global statistics or just straight correlation. But cosine distance (which adds the scaling operation) decreases performance substantially for the low recall region. In general, removing the mean response is far more important than scaling by the standard deviation. Interestingly, in the case of cosine distance and global channel normalization, scaling by the standard deviation actually hurts performance (*i.e.*, $[\mu, \sigma]$ versus $[\mu, \cdot]$ and $[\bar{\mu}_c, \bar{\sigma}_c]$ versus $[\bar{\mu}_c, \cdot]$ respectively). As normalization re-weights channels, we posit that this may be negatively effecting the scores by down-weighing important signals or boosting noisy signals.

Channel decorrelation: Recall that, for efficiency reasons, our multivariate estimate of correlation assumes that channels are largely decorrelated. We also explored decorrelating the channels globally using a full-dimension PCA (which also subtracts out the global mean $\bar{\mu}_c$). The right panel of Fig. 4.4 shows a comparison of these decorrelated feature channels (solid curves) relative to baseline ResNet channels (dotted curves). While the decorrelated

features outperform baseline correlation (due to the mean subtraction) we found that full MCNCC on the raw features performed better than on globally decorrelated features. This may be explained in part due to the fact that decorrelated features show an even wider range of variation across different channels which may exacerbate some of the negative effects of scaling by σ_c.

Other feature extractors: To see if this behavior is specific to the ResNet-50 model, we evaluate on three additional features: raw pixels, GoogleNet, and DeepVGG-16. From the GoogleNet model[2] we used the 192-channel 'conv2x' activations, and from the DeepVGG-16 model[3] we used the 256-channel 'x12' activations. We chose these particular CNN feature maps because they had the same or similar spatial resolution as 'res2bx' and were the immediate output of a rectified linear unit layer.

As shown in Table 4.1, we see a similar pattern to what we observed with ResNet-50's 'res2bx' features. Namely, that straight cross-correlation (denoted as $[\cdot, \cdot]$) performs poorly, while MCNCC (denoted as $[\mu_c, \sigma_c]$) performs the best. One significant departure from the previous results for 'res2bx' features is how models using entire feature volume statistics perform. Centering using 3D statistics (denoted as $[\mu, \cdot]$) yields performance that is closer to straight correlation, on the other hand, standardizing using 3D statistics (denoted as $[\mu, \sigma]$) yields performance that is closer to MCNCC when using GoogleNet's 'conv2x' and DeepVGG-16's 'x12' features.

When we look at the difference between the per-channel and the across-channel (3D) statistics for query patches, we observe significant difference in sparsity of μ_c compared to μ: 'conv2x' is about 2x more sparse than 'x12,' which itself is about 2x more sparse than 'res2bx.' The level of sparsity correlates with the performance of $[\mu, \cdot]$ compared to straight

[2]Pretrained model was obtained from `http://www.vlfeat.org/matconvnet/models/imagenet-googlenet-dag.mat`

[3]Pretrained model was obtained from `http://www.vlfeat.org/matconvnet/models/imagenet-vgg-verydeep-16.mat`

Features	$[\cdot, \cdot]$	$[\mu, \cdot]$	$[\mu, \sigma]$	$[\mu_c, \cdot]$	$[\mu_c, \sigma_c]$
Raw Pixels	0.04	0.20	0.45	-	-
ResNet-50 (res2bx)	0.15	0.44	0.32	0.55	0.77
GoogleNet (conv2x)	0.07	0.09	0.68	0.61	0.81
DeepVGG-16 (x20)	0.09	0.31	0.73	0.51	0.76

Table 4.1: Ablation study on the two normalized cross-correlation schemes across different features. We measure performance using mean average precision, higher is better. As the images are gray-scale single-channel images, for raw pixels $[\mu, \cdot]$ and $[\mu, \sigma]$ are identical to $[\mu_c, \cdot]$ and $[\mu_c, \sigma_c]$, respectively.

correlation across the different features. The features where μ_c is more sparse, using μ over-shifts across more channels leading to less performance gain relative to straight correlation. When we look at the difference between σ and σ_c, we observe that σ is on average larger than σ_c. This means that compared to σ_c, using σ dampens the effect of noisy channels rather than boosting them. Looking at the change of performance from $[\mu, \cdot]$ to $[\mu, \sigma]$ for different features, we similarly see improvement roughly correlates to how much larger σ is than σ_c.

4.6 Cross-Domain Matching Experiments

In this section, we evaluate our proposed system in settings that closely resembles various real-world scenarios where query images are matched to a database containing images from a different domain than that of the query. We focus primarily on matching crime scene prints to a collection of test impressions, but also demonstrate the effectiveness of MCNCC on two other cross-domain applications: semantic segmentation label retrieval from building facade images, and map retrieval from aerial photos.[4] As in our diagnostic experiments, we use the same pre-trained ResNet-50 model. We use the 256-channel 'res2bx' activations for the shoeprint and building facade data, but found that the 1024-channel 'res4cx' activations performed better for the map retrieval task.

[4]Our code is available at `http://github.com/bkong/MCNCC`

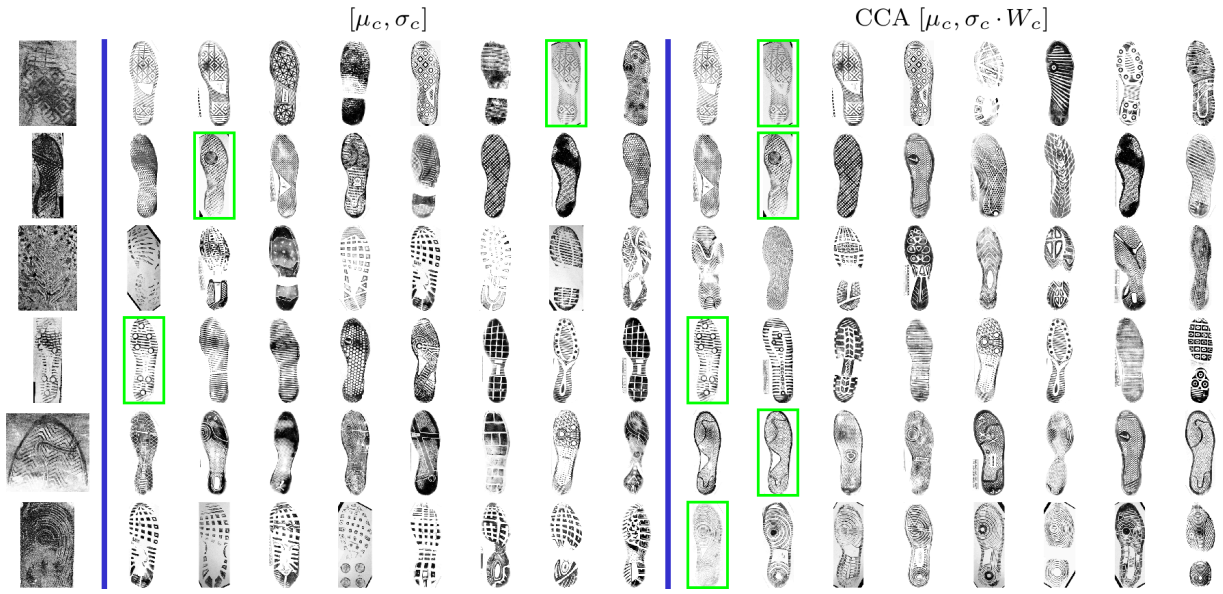


Figure 4.6: FID-300 retrieval results. The left column shows the query crime scene prints, the middle column shows the top-8 results for $[\mu_c, \sigma_c]$, and the right column shows the top-8 results for CCA $[\mu_c, \sigma_c \cdot W_c]$. Green boxes indicate the corresponding ground truth test impression.

4.6.1 Shoeprint Retrieval

In addition to the internal dataset described in Section 4.5, we also evaluated our approach on a publicly available benchmark, the footwear identification dataset (FID-300) [35]. FID-300 contains 1175 test impressions and 300 crime scene prints. The task here is similar to the diagnostic experiments on patches, but now matching whole prints across domains. As the crime scene prints are not aligned to a canonical orientation, we search over both translations (with a stride of 2) and rotations (from -20° to +20° with a stride of 4°). For a given alignment, we compute the valid support region P where the two images overlap. The local statistics and correlation is only computed within this region.

As mentioned in Sec. 4.4, we can learn both the linear projections of the features and the importance of each channel for the retrieval task. We demonstrate that such learning is feasible and can significantly improve performance. We use a 50/50 split of the crime scene prints of the Israeli dataset for training and testing, and determine hyperparameters settings

using 10-fold cross-validation. In the left panel of Fig. 4.5 we compare the performance of three different models with varying degrees of learning. The model with no learning is denoted as $[\mu_c, \sigma_c]$, with learned per-channel weights is denoted as $[\mu_c, \sigma_c \cdot W_c]$, with learned projections is denoted as CCA $[\mu_c, \sigma_c]$, and with piece-wise learned linear projections and per-channel weights is denoted as CCA $[\mu_c, \sigma_c \cdot W_c]$. Our final model, CCA $[\mu_c, \sigma_c \cdot W_c]$ ft, jointly fine-tunes the linear projections and the per-channel weights together. The model with learned per-channel importance weights has 257 parameters (a scalar for each channel and a single bias term), and was learned using a support vector machine solver with a regularization value of $\alpha = 100$. The linear projections (CCA) were learned using `canoncorr`, MATLAB's canonical correlation analysis function. Our final model, CCA $[\mu_c, \sigma_c \cdot W_c]$ ft, was fine-tuned using gradient descent with an L2 regularization value of $\alpha = 100$ on the per-channel importance weights and $\beta = 1$ on the linear projections. This full model has 131K parameters (2×256^2 projections, 256 channel importance, and 1 bias).

As seen in the left panel of Fig. 4.5, learning per-channel importance weights, $[\mu_c, \sigma_c \cdot W_c]$, yields substantial improvements, outperforming $[\mu_c, \sigma_c]$ and CCA $[\mu_c, \sigma_c]$ when recall is less than 0.34. When learning both importance weights and linear projections, we see gains across all recall values as our Siamese network significantly outperforms all other models. However, we observe only marginal gains when fine-tuning the whole model. We expect this is due in part to the small amount of training data which makes it difficult to optimize parameters without overfitting.

We subsequently tested these same models (without any retraining) on the FID-300 benchmark (shown in the right panel of Fig. 4.5). In this, and in later experiments, we use cumulative match characteristic (CMC) which plots the percentage of correct matches (recall) as a function of the number of database items reviewed. This is more suitable for performance evaluation than other information retrieval metrics such as precision-recall or precision-at-k since there is only a single correct matching database item for each query.

CMC is easily interpreted in terms of the actually use-case scenario (i.e., how much effort a forensic investigator must expend in verifying putative matches to achieve a given level of recall).

On FID-300, we observe the same trend as on the Israeli dataset — models with more learned parameters perform better. However, even without learning (*i.e.*, $[\mu_c, \sigma_c]$) MCNCC significantly outperforms using off-the-shelf CNN features the previously published state-of-the-art approaches of Kortylewski et al. [35, 36, 34] The percentage of correct matches at top-1% and top-5% of the database image reviewed for ACCV are 14.67 and 30.67, for BMVC16 are 21.67 and 47.00, for LoG16 are 59.67 and 73.33, for $[\mu_c, \sigma_c]$ are 72.67 and 82.33, and for CCA $[\mu_c, \sigma_c]$ ft are 79.67 and 86.33. In Fig. 4.6, we visualize the top-10 retrieved test impressions for a subset of crime scene query prints from FID-300. These results correspond to the CMC curves for $[\mu_c, \sigma_c]$ and CCA $[\mu_c, \sigma_c \cdot W_c]$ of the right panel of Fig. 4.5.

Partial occlusion: To analyze the effect of partial occlusion on matching accuracy, we split the set of crime scene query prints into subsets with varying amounts of occlusion. For this we use the proxy of pixel area of the cropped crime scene print compared to its corresponding test impression. The prints were then grouped into 4 categories with roughly equal numbers of examples: "Full size" prints are those whose pixel-area ratios fall between $[0.875, 1]$, "3/4 size" between $[0.625, 0.875)$, "half size" between $[0.375, 0.625)$, and "1/4 size" between $[0, 0.375)$. In Table 4.2 we compare the performance of models $[\mu_c, \sigma_c]$, CCA $[\mu_c, \sigma_c]$, and CCA $[\mu_c, \sigma_c \cdot W_c]$. As expected, the correct match rate generally increases for all models as the pixel area ratio increases and more discriminative tread features are available, with the exception of "full size" prints. While "full size" query prints might be expected to include more relevant features for matching, we have observed that in the benchmark dataset they are often corrupted by additional "noise" in the form of smearing or distortion of the print and marks left by overlapping impressions.

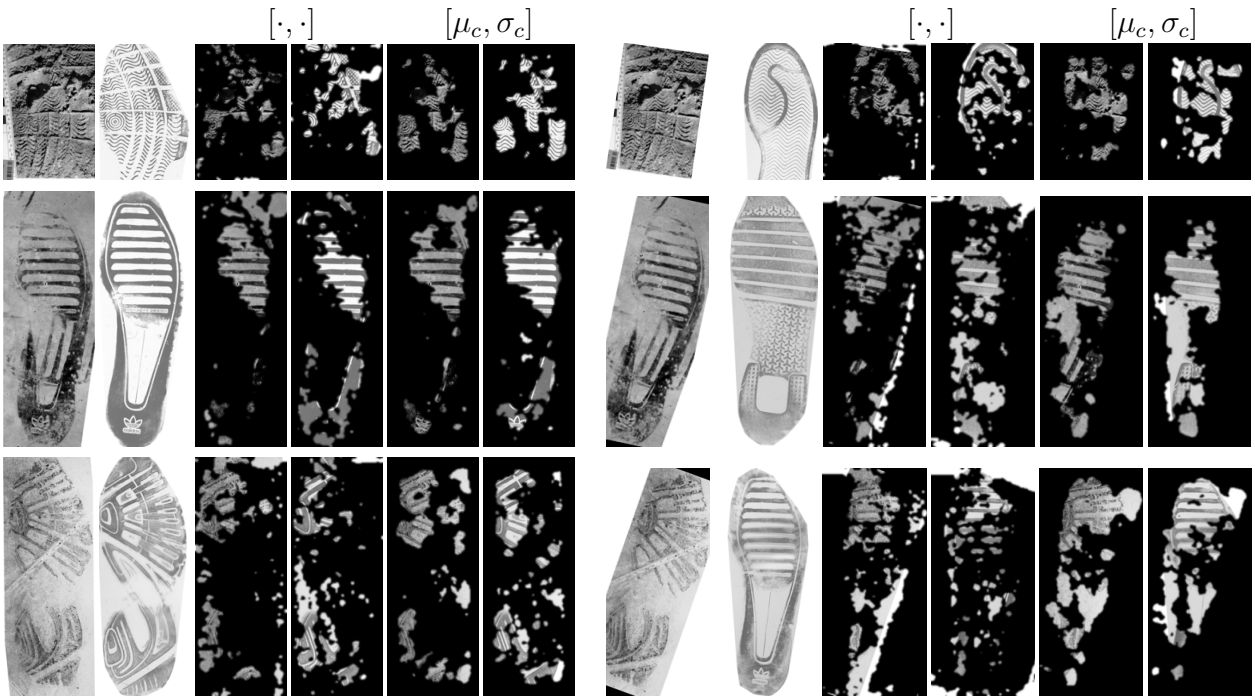


Figure 4.7: Visualizing image regions that have the greatest influence on positive correlation between image pairs. Each group of images shows, from left to right, the original crime scene print and test impression being compared, the image regions of the pair that have the greatest influence on positive correlation score when using raw cross-correlation, and the image regions of the pair that have greatest influence on positive MCNCC. Each row shows the same crime scene query aligned with a true matching impression (left) and with a non-matching test impression (right).

Background clutter: We also examined how performance was affected by the amount of irrelevant background clutter in the crime scene print. We use the ratio of the pixel area of the cropped crime scene print over the pixel area of the original crime scene print as a proxy for the amount of relevant information in a print. Prints with a ratio closer to zero contain a lot of background, while prints with a ratio closer to one contain little irrelevant information. We selected 257 query prints with a large amount of background (ratio ≤ 0.5).

When performing matching over these whole images we found that the percentage of correct top-1% matches dropped from 72.4% to 15.2% and top-10% dropped from 88.3% to 33.5%. This drop in performance is not surprising given that our matching approach aims to answer the question of *what* print is present, rather than detecting *where* a print appears in an image

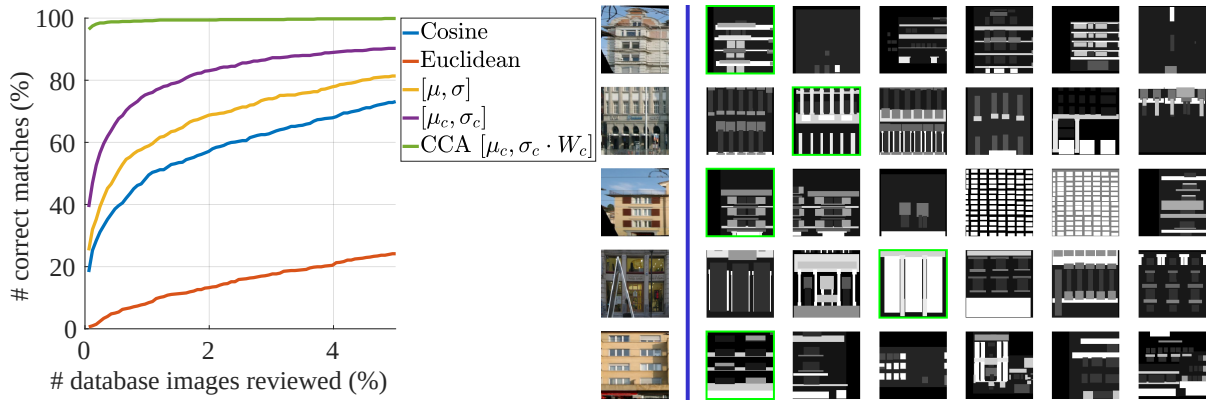


Figure 4.8: Segmentation retrieval for building facades. The left panel compares MCNCC with learned linear projections and per-channel importance weights (denoted as CCA $[\mu_c, \sigma_c \cdot W_c]$) and MCNCC with no learning (denoted as $[\mu_c, \sigma_c]$) to other baseline metrics: Cosine similarity, Euclidean distance, and NCC using across-channel local statistics (denoted as $[\mu, \sigma]$). The right panel shows example retrieval results for CCA $[\mu_c, \sigma_c \cdot W_c]$. The left column shows the query facade image. Green boxes indicate the corresponding ground truth segmentation label.

and was not trained to reject background matches. We note that in practical investigative applications, the quantity of footwear evidence is limited and a forensic examiner would likely be willing to mark valid regions of query image, limiting the effect of background clutter.

Visualizing image characteristics relevant to positive correlations: To get an intuitive understanding of what image features are utilized by MCNCC, we visualize what image regions have a large influence the positive correlation between paired crime scene prints and test impressions. For a pair of images, we backpropagate gradients to the image from each spatial bin in the feature map which has a positive normalized correlation. We then produce a mask in the image domain marking pixels whose gradient magnitudes are in the top 20th percentile. Fig. 4.7 compares this positive relevance map for regular correlation (inner product of the raw features) and normalized correlation (inner product of the standardized features). We can see that with normalized correlation, the image regions selected are similar for both images despite the domain shift between the query and match. In contrast, the visualization for regular correlation shows much less coherence across the pair of images and

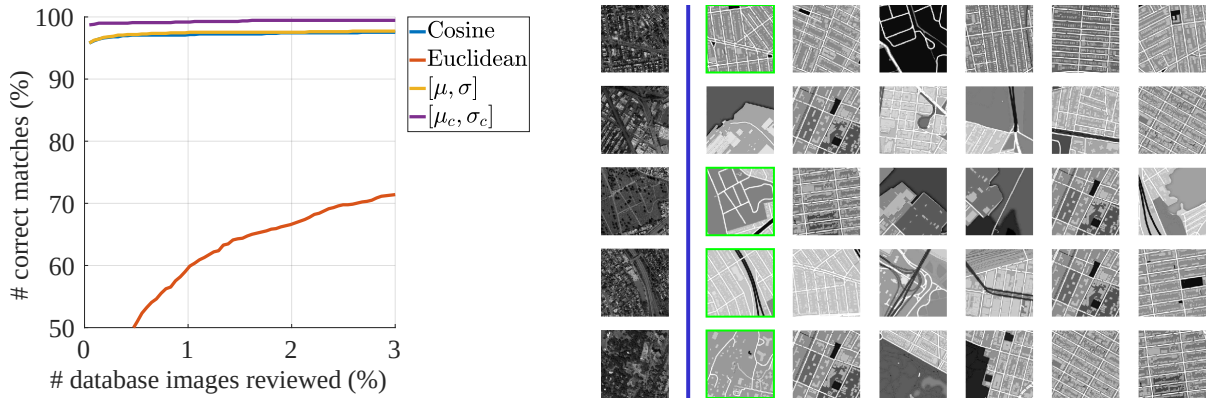


Figure 4.9: Retrieval of maps from aerial imagery. The left panel compares MCNCC with no learning (denoted as $[\mu_c, \sigma_c]$) to other baseline metrics: Cosine similarity, Euclidean distance, and NCC using across-channel per-exemplar statistics (denoted as $[\mu, \sigma]$). The right panel shows retrieval results for $[\mu_c, \sigma_c]$. The left column shows the query aerial photo. Green boxes indicate the corresponding ground-truth map image.

often attends to uninformative background edges and blank regions.

4.6.2 Segmentation Retrieval for Building Facades

To further demonstrate the robustness of MCNCC for cross domain matching, we consider the task of retrieving segmentation label maps which match for a given building facade query image. We use the CMP Facade Database [52] which contains 606 images of facades from different cities around the world and their corresponding semantic segmentation labels. These labels can be viewed as a simplified "cartoon image" of the building facade by mapping each label to a distinct gray level.

In our experiments, we generate 1657 matching pair by resizing the original 606 images (base + extended dataset) to either 512×1536 or 1536×512 depending on their aspect ratio and crop out non-overlapping 512×512 patches. We prune this set by removing 161 patches which contain more than 50% background pixels to get our final dataset. Examples from this dataset can be seen in the right panel of Fig. 4.8. In order treat the segmentation label

		all prints	full size	3/4 size	half size	1/4 size
# prints		300	88	78	71	63
Top-1%	$[\mu_c, \sigma_c]$	72.7	78.4	82.1	71.8	53.0
	CCA $[\mu_c, \sigma_c]$	76.8	83.0	85.9	73.2	60.3
	CCA $[\mu_c, \sigma_c \cdot W_c]$	79.0	84.1	85.9	78.9	63.5
Top-10%	$[\mu_c, \sigma_c]$	87.7	87.5	92.3	85.9	84.1
	CCA $[\mu_c, \sigma_c]$	88.7	93.2	91.0	87.3	81.0
	CCA $[\mu_c, \sigma_c \cdot W_c]$	89.3	93.2	91.0	91.6	79.4

Table 4.2: Occlusion study on FID-300. The crime scene query prints are binned by looking at the ratio of query pixel area to the pixel area of the corresponding ground-truth test impression. Performance is measured as the percentage of correct matches retrieved (higher is better).

map as an image suitable for the pre-trained feature extractor, we scale the segmentation labels to span the whole range of gray values (*i.e.*, from $[1 - 12]$ to $[0 - 255]$).

We compare MCNCC (denoted in the legend as $[\mu_c, \sigma_c]$) to three baseline similarity metrics: Cosine, Euclidean distance, and normalized cross-correlation using across-channel local statistics (denoted as $[\mu, \sigma]$). We can see in the left panel of Fig. 4.8 that MCNCC performs significantly better than the baselines. MCNCC returns the true matching label map as the top scoring match in 39.2% of queries. In corresponding top match accuracy for normalized cross-correlation using across-channel local statistics is 25.2%, for Cosine similarity is 18.3%, and for Euclidean distance is 6.0%. When learning parameters with MCNCC (denoted as CCA $[\mu_c, \sigma_c \cdot W_c]$), using a 50/50 training-test split, we see significantly better retrieval performance (96.4% for reviewing one database item). The right panel of Fig. 4.8 shows some example retrieval results for this model.

4.6.3 Retrieval of Maps from Aerial Imagery

Finally, we evaluate matching performance on the problem of retrieving map data corresponding to query aerial photos. We use a dataset released by Isola *et al.* [24] that contains

2194 pairs of images scraped from Google Maps. For simplicity in treating this as a retrieval task, we excluded map tiles which consisted entirely of water. Both aerial photos and map images were converted from RGB to gray-scale prior to feature extraction (see the right panel of Fig. 4.9 for examples). We compare MCNCC to three baseline similarity metrics: Cosine, Euclidean distance, and normalized cross-correlation using across-channel local statistics (denoted as $[\mu, \sigma]$).

The results are shown in the left panel of Fig. 4.9. MCNCC outperforms the baseline Cosine and Euclidean distance measures, but this time performance of normalized cross-correlation using local per-exemplar statistics averaged over all channels and Cosine similarity are nearly identical. For top-1 retrieval performance, MCNCC is correct 98.7% of the time, normalized cross-correlation using across-channel local statistics and Cosine similarity are correct 95.8%, and Euclidean distance is correct 28.6% of the time when retrieving only one item. We show example retrieval results for MCNCC in the right panel of Fig. 4.9. We did not evaluate any learned models in this experiment since the performance of baseline MCNCC left little room for improvement.

4.7 Conclusion

In this work, we proposed an extension to normalized cross-correlation suitable for CNN feature maps that performs normalization of feature responses on a per-channel and per-exemplar basis. The benefits of performing per-exemplar normalization can be explained in terms of spatially local whitening which adapts to non-stationary statistics of the input. Relative to other standard feature normalization schemes (*e.g.*, cosine similarity), per-channel normalization accommodates variation in statistics of different feature channels.

Utilizing MCNCC in combination with CCA provides a highly effective building block for

constructing Siamese network models that can be trained in an end-to-end discriminative learning framework. Our experiments demonstrate that even with very limited amounts of data, this framework achieves robust cross-domain matching using generic feature extractors combined with piece-wise training of simple linear feature-transform layers. This approach yields state-of-the art performance for retrieval of shoe tread patterns matching crime scene evidence. We expect our findings here will be applicable to a wide variety of single-shot and exemplar matching tasks using CNN features.

Chapter 5

Alignment Prediction for Fast Image Matching

In the previous chapter, we looked at a cross-domain shoeprint matching pipeline that worked well but was suboptimal—in the sense that it needs to search over possible alignments (rotations and translations) to maximize the correlation score between a query crime scene print and a database laboratory test impression. In this chapter, we address this issue by leveraging the fact that tread patterns from the same make/model shoe are largely uniform in appearance and that they are rigid in shape, in order to learn a model that can predict the optimal alignment directly without performing search.

5.1 Introduction

Object misalignment is a common problem in image matching. Even for rigid objects, like shoes, their appearance from one image to another can change depending on how the object is captured (*e.g.*, difference in viewpoint or orientation). One strategy to address misalignment

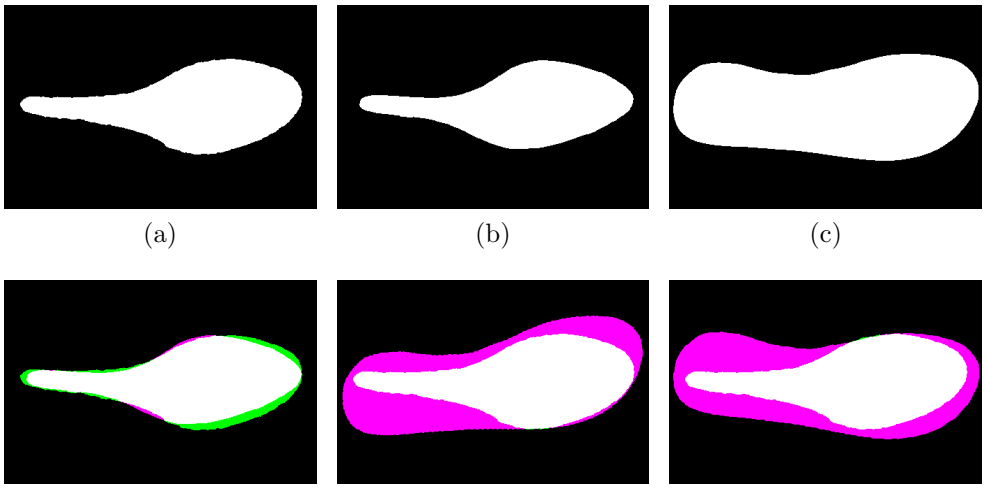
is to explicitly search over a set of transformations to bring the object in different images into alignment (as seen in Chapter 4). This is optimal in terms of matching, but is resource intensive and slow. Another strategy is to predict the alignment between two images—which is the strategy we explore in this work.

For image retrieval, if the optimal alignment of a query to each database item were different (because items in the database are in various poses), then performing a database search requires us to predict a separate alignment for each database item—making a search at best linear time w.r.t.the number of database items. The ideal solution to this problem would be to align the database into a canonical pose—allowing us to predict a single alignment of the query to the entire database. We can formally define retrieving the most similar database item as

$$\mathrm{S}(\mathbf{q}) = \arg\max_{\mathbf{d}^* \in \mathcal{D}} \mathrm{MCNCC}(f_\theta(\mathbf{q}), \mathbf{d}^*), \tag{5.1}$$

where $\mathbf{q}$ is a query image, $\mathcal{D}$ is the database, $\mathbf{d}^*$ is a database item aligned to a canonical pose, $f_\theta(\cdot)$ is a function that transforms its input to the canonical pose (alignment predictor), and $\mathrm{MCNCC}(\cdot, \cdot)$ is the learned similarity metric described in Chapter 4. Transforming an image to a canonical pose is in the same vein as learning features that are invariant to geometric transformations [25], however, by operating on the image we can use any existing feature extractor or similarity metric. Additionally, with the database aligned computing the similarity a query and a database item using the MCNCC metric is simply the inner product between (standardized) vectors, which is amenable to acceleration (see *e.g.*, [1, 28]).

A problem with defining a canonical pose across all shoes is that it is ill-defined (see illustration in Fig. 5.1). Instead we propose a middle-ground approach and align the database to a set of poses, where the size of the set is determined by the number of dissimilar shoe shapes in the database—which we would choose to be smaller than the number of database



(a) (b) (c)

(d) Aligning (a) with (b) (e) Aligning (a) with (c) (f) Aligning (a) with (c)

Figure 5.1: Aligning shoes by their shape. The first row shoes the shapes of three different shoes. The second row shows possible alignments. (a) and (b) are similar in shape, so there is only a single reasonable alignment (d), while (a) and (c) are dissimilar in shape, so there are multiple reasonable alignments (e) and (f).

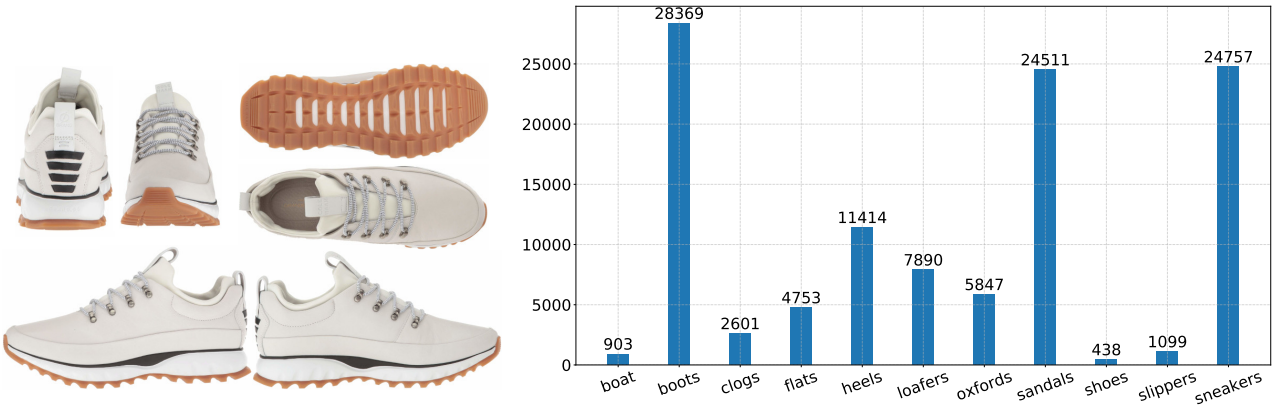
items (*e.g.*, $O(\log|\mathcal{D}|)$) in order to get sublinear search. We can then predict the alignment between the query and a reference pose, and search the database using the set of alignments. Updating Eq. 5.1 to the proposed idea gives us

$$\mathrm{S}(\mathbf{q}) = \arg\max_{\mathbf{d}^* \in \mathcal{D}} \max_{\mathbf{r} \in \mathcal{R}} \mathrm{MCNCC}(f_\theta(\mathbf{q}, \mathbf{r}), \mathbf{d}^*), \tag{5.2}$$

where we now include $\mathbf{r} \in \mathcal{R} \subset \mathcal{D}$ a reference image and change $f_\theta(\cdot, \cdot)$ to transform its first input to the pose of its second input.

5.1.1 SHoe Outsole Dataset (UCI-SHOD)

We briefly introduce our new shoe outsole dataset (UCI-SHOD). Our dataset includes shoe images from six different viewpoints: bottom, top, left, right, front, and back (see Fig. 5.2a for example images), but we focus on just the bottom images in this work. It also includes meta-data about shoes: gender (male, female, unisex), brand, product name, and shoe



(a) Example images

(b) Distribution of shoes across categories

category (boat shoe, boots, clogs and mules, flats, heels, loafers, oxfords, sandals, slippers, sneakers and athletic shoes, and other shoes). We collect this data (on a monthly basis) by scraping the websites of online shoe retailers: OnlineShoes, 6pm.com, and Zappos.com. And have thus far we have collected 288K outsole images across 112K different shoes, and continue to collect about 5K new shoes per month. The average number of outsole images per shoe is 2.56, with a median of 2.0, and a max of 203. In Fig. 5.2b we show the distribution of shoes across the shoe categories.

5.2 Transform Parameterization

For simplicity, we consider only the set of transformations that are affine. We parameterize this set by composing a transform from the following component transforms: x-translation, y-translation, scale, rotation angle, shear angle. When given a parameter vector $\mathbf{v} = (t_x, t_y, s, \theta, \rho)$, a transformation matrix A is formed by multiplying component matrices

in the following fixed order:

$$\begin{aligned}\mathbf{A} &= F(t_x, t_y, s, \theta, \rho) \\ &= \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin(\theta+\rho) & 0 \\ \sin\theta & \cos(\theta+\rho) & 0 \\ 0 & 0 & 1 \end{bmatrix}\end{aligned}$$

5.2.1 Alignment Network

We use a spatial transformer network (STN) [25] to predict affine parameters for transforming a query to align with a reference. STNs were introduced as a way to given neural networks the ability to spatially transform feature maps—conditioned only on the feature maps—without any extra training supervision. They are compose of three elements: a localization network that outputs the parameters of the transformation to be applied to the input, a grid generator that creates a mapping of the inputs that should be sampled for each output location, and a sampler that performs interpolation operation. We propose to use a STN as a stand-alone network that *is* conditioned on a reference image as described in Eq. 5.2. And while STNs can be used for more general transformations (*e.g.*, plane projective transformations or thin plate spline), we constrain the set of transformations to be affine.

5.3 Database Alignment

We approach the problem of aligning an image database of shoes by aligning sets of images that belong to each shoe. Bringing a set of images into alignment ("congealing") has been used in many different applications: building brain atlases [31], boosting face and object recognition accuracy [23], and removing systematic measurement noise from brain scans [39].

Figure 5.2: Shoe images from the same class can have different appearances due to other parts of the shoe being captured in addition to the outsole. The first row shows the original images, where the side of the shoe and the laces may cause problems when trying to align using pixels. The second row shows the same images with the extraneous parts masked out, where alignment using pixels will not be a problem.

We bring congealing to the application of accelerating image retrieval. We use a two stage process: initially aligning using shape, then aligning using both shape and pixels.

5.3.1 Shapes

Shape is essential for aligning across classes where we expect tread patterns to be different, but is useful for aligning within classes. The same shoe can appear quite differently across images due to the acquisition method, which may produce shadows and expose other extraneous parts of the shoe in addition to the outsole (see the first row of Fig. 5.2). However, we can make their appearance more consistent by masking out these parts of the image (as illustrated in the second row of Fig. 5.2). We might think that using only shape should suffice in aligning within class images, however, as with aligning using only appearance (pixels) we expect the shapes to be imperfect—motivating us to do the second round of alignment using both shape and raw pixels.

To generate shape masks, we use a pyramid scene parsing network (PSPNet) [77]. PSPNets capture both local and global contextual information for image segmentation by building a feature pyramid that is used to make pixel-level predictions. The levels of the feature pyramid are computed much like an image pyramid (through a series of filtering and downsampling operations), but with learned filters. Levels of the pyramid are combined by upsampling and concatenating each level of the feature pyramid with the original feature map, which is then used to make pixel-level predictions. We use a PSPNet to make a binary prediction of outsole versus non-outsole.

5.3.2 Procedure

We now discuss the exact procedure for aligning a set of images (and is summarized in Alg. 2). We do this in two stages: the first is a coarse alignment using shape and the second is a fine alignment using shape and raw pixels. The two stages are otherwise identical. We use the following notations: $\mathbf{x}_j$ is an image from the set being aligned, S is a shape generator (*e.g.*, a pyramid scene parsing network), P is an alignment predictor (*e.g.*, a spatial transformer network), and E is a feature extractor. In the first stage $E(\mathbf{x}) = S(\mathbf{x})$, however, in the second stage $E(\mathbf{x})$ outputs the concatenation of $S(\mathbf{x})$ and $\mathbf{x}$.

We begin the alignment procedure by choosing a reference image to align the set to, denoted as $\bar{\mathbf{x}}$. We use the medoid of the set because we want the set of transformations that align the images to the reference image to be as small as possible, however, a reference can also be randomly selected from the set. We believe the only poor choice would be to use a reference that was computed from averaging all the images in the set—the set is not yet aligned and so the result will likely be a blurred image.

Next we align the set to the reference image using the alignment predictor. There are two things to note about aligning an image to the reference. Firstly, we use the alignment

predictor recursively, transforming $\mathbf{x}_j$ until the predicted alignment $\mathbf{v}_j$ from P is near the identity transformation (convergence criteria). Secondly, rather than transform the previous transformed image, we accumulate the transformations and always transform the original image—which minimizes artifacts from interpolation and preserves more high-frequency information.

Finally, from the set of aligned images we determine the class shape mask, used for computing the similarity between classes, through consensus voting—computing the average shape mask over the set and determining foreground outsole where more than half the images agree. We use this voting scheme to determine the class shape as it can help mitigate possible errors made by shape generator. From the class shape mask we compute the shift to center the shape to the middle of the image. We add this final transformation to the accumulated transform $\mathbf{A}_j$ to transform $\mathbf{x}_j$ one last time to get our aligned set.

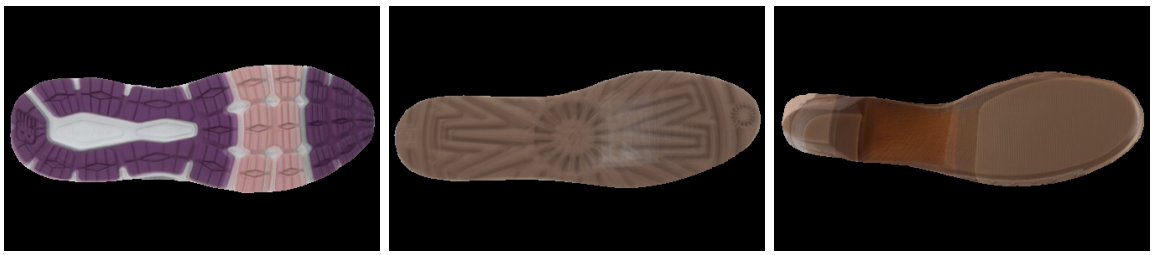


Figure 5.3: Per-class mean image to show within class alignment quality. Sharper tread patterns indicate better alignment.

Algorithm 2 Procedure for Aligning Set of Images

Choose reference from set $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \rightarrow \bar{\mathbf{x}}$

Compute reference features $E(\bar{\mathbf{x}}) \rightarrow \bar{\mathbf{z}}$

Align set to reference:

for $j = 1$ to N **do**

    $\mathbf{A}_j = F(0, 0, 1, 0, 0)$

    do

        Apply transformation $\mathbf{A}_j$ to $\mathbf{x}_j \rightarrow \hat{\mathbf{x}}_j$

        Compute image features $E(\hat{\mathbf{x}}_j) \rightarrow \hat{\mathbf{z}}_j$

        Predict transformation parameters $P(\hat{\mathbf{z}}_j, \bar{\mathbf{z}}) \rightarrow \mathbf{v}_j$

        $F(\mathbf{v}_j)\mathbf{A}_j \rightarrow \mathbf{A}_j$

    while $\mathbf{v}_j$ not converged

end for

Consensus voting for class shape:

Initialize $\bar{\mathbf{s}}$

for $j = 1$ to N **do**

    Apply transformation $\mathbf{A}_j$ to $\mathbf{x}_j \rightarrow \hat{\mathbf{x}}_j$

    $S(\hat{\mathbf{x}}_j) \rightarrow \mathbf{s}_j$

    $\bar{\mathbf{s}} + \frac{1}{N}\mathbf{s}_j \rightarrow \bar{\mathbf{s}}$

end for

Threshold $\bar{\mathbf{s}} \geq \frac{1}{2} \rightarrow \bar{\mathbf{s}}$

Global centering by class shape:

Compute parameters to center $\bar{\mathbf{s}} \rightarrow \mathbf{v}_{\bar{\mathbf{s}}}$

for $j = 1$ to N **do**

    $F(\mathbf{v}_{\bar{\mathbf{s}}})\mathbf{A}_j \rightarrow \mathbf{A}_j^*$

    Apply transformation $\mathbf{A}_j^*$ to $\mathbf{x}_j \rightarrow \mathbf{x}_j^*$

end for

Training: To align the class images according the procedure discussed in the previous section we train two models: a shape generator (pyramid scene parsing network described in Sec. 5.3.1) and an alignment predictor (spatial transformer network described in Sec. 5.2.1).

The shape generator is trained using 999 randomly selected images across all classes, where the images have been annotated to indicate which pixels are outsole and non-outsole. We follow the same optimization procedure described in [77] (same loss function and learning rates).

The alignment predictor is trained using the same set of images used to train the shape generator using self-matching pairs. For both stages, we train the predictor using self-matching pairs—a pair with the original image and the transformed version of the image. The original image is used as the reference and target, while the transformed image is used as the input. We transform the image by randomly generated parameters: x-translation between $(-0.02 * \text{width}, 0.02 * \text{width})$, y-translation between $(-0.1 * \text{height}, 0.1 * \text{height})$, scale between $(0.95, 1.05)$, rotation degrees between $(-10, 10)$, and skew degrees between $(-1, 1)$. We train two different models to correspond to the two stages of the alignment process: shape only and shape with pixels. When training the model for shape only, we augment the training data by randomly choose between using the ground truth mask and the mask from the shape generator. When training the model for shape with pixels, we use only the mask from the shape generator.

5.3.3 Results

While difficult to quantify how well our class alignment procedure performs, it is quite easy to do qualitatively. Fig. 5.3 shows the mean aligned image of four different classes; We can see that images in most classes are able to align fairly well, however, there are failure cases like the bottom-left panel of Fig. 5.3. Upon inspection we see that most low quality alignments

Figure 5.4: Out-of-plane rotation is the main cause of poor alignment for within class images. Left panel is the same average image shown in last panel of Fig. 5.3, other panels show original unaligned images for that class.

are caused by out-of-plane rotation (see Fig. 5.4), which cannot be resolved through an affine transformation that only supports in-plane rotations.

5.4 Experiments

In this section, we evaluate our alignment predictor in an experimental setting where query images are held out shoe images from the database. We use a dataset that consists of 26 classes and 1198 whole shoe images (448×292 resolution), which we split the images in each class 70% for training and 30% for testing. From each image in the test set, we synthesize 4 additional images by cropping the shoe, centering the crop in the middle of the image, and randomly transformed by scale, rotation, and skew parameters (see "Input" in Fig. 5.5). Patch sizes (computed as shoe area in the crop over shoe area in the original image) are: 0.333, 0.5, 0.667, and 0.833. With the additional images, the testing set totals to 1245 images (from 249 unique images).

Training: Similar to the alignment predictor used in Sec. 5.3, we use self-matching pairs to train the model. The original unaltered image is used as the reference, a cropped version (to simulate occlusion) of the image can be used as the target, and a version of the target image where the shoe content is centered in the image and is randomly transformed by scale, rotation, and skew parameters is used as the input (see Fig. 5.5 for an example). The size

(a) Input (b) Target (c) Reference

Figure 5.5: Example used for training the alignment predictor. (Black border added for ease of viewing patch pose in image.)

of the cropped patch follows a uniform distribution from three tenths the size of shoe to the full shoe. We train the predictor with a loss on the predicted parameters and the inverted parameters used to transform the target image to the input image. Although training using a loss on the transformed input image and target image would be just as reasonable, training on the parameters is more precise and should not hurt performance.

5.4.1 Alignment Prediction

We first look at the accuracy of the parameters when aligning two images from the same make/model shoe. This is particularly relevant as, recall from Eq. 5.2, we search over a set of alignments by aligning a query to a set of references as alignment across classes is an ill-posed problem. In Table 5.1 we show the difference between the predicted parameters and the ground truth parameters for different size queries in the test set. We can see the error of each parameter is fairly uniform across the different sizes. The exception is x-translation (t_x) where the error for a patch size of 1.0 is three fifths that of the other sizes; this is unsurprising as there is much less room to shift the shoe along the x-axis. However, when considering relative error (Table 5.2), we see for patch size of 1.0 the t_x parameter is higher than the other sizes.

We can also consider how prediction can alleviate the burden of explicit alignment search

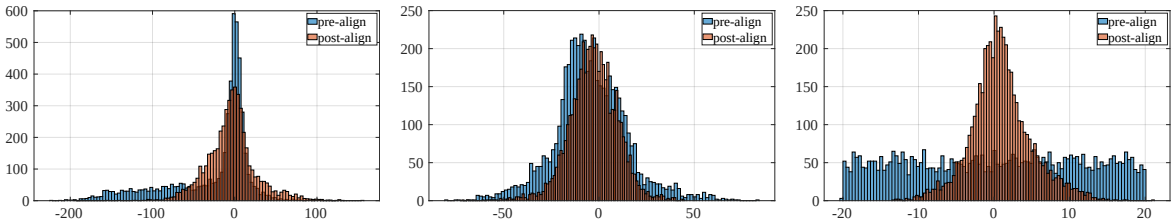
patch size	t_x (pix)	t_y (pix)	scale	rot (deg)	skew (deg)
1.0	14 ± 13	10 ± 7	0.02 ± 0.02	3.84 ± 3.07	0.49 ± 0.29
0.833	25 ± 22	10 ± 8	0.02 ± 0.02	2.90 ± 2.70	0.50 ± 0.29
0.667	25 ± 22	10 ± 8	0.02 ± 0.02	2.82 ± 2.59	0.51 ± 0.29
0.5	24 ± 23	10 ± 9	0.02 ± 0.02	2.99 ± 2.75	0.50 ± 0.30
0.333	24 ± 21	10 ± 9	0.02 ± 0.02	2.88 ± 2.69	0.50 ± 0.30

Table 5.1: Absolute error of predicted parameters.

patch size	t_x	t_y	scale	rot	skew
1.0	4.22 ± 7.05	3.80 ± 25.59	0.02 ± 0.02	0.94 ± 4.19	1.32 ± 8.79
0.833	2.58 ± 5.26	4.01 ± 33.44	0.02 ± 0.01	1.32 ± 11.63	1.06 ± 0.60
0.667	2.61 ± 5.08	3.75 ± 43.06	0.02 ± 0.02	0.97 ± 6.95	1.46 ± 10.09
0.5	2.80 ± 7.38	2.71 ± 23.36	0.02 ± 0.02	0.70 ± 2.59	1.08 ± 0.95
0.333	2.54 ± 5.29	3.75 ± 42.71	0.02 ± 0.02	0.81 ± 4.56	1.20 ± 3.57

Table 5.2: Relative error of predicted parameters.

rather than replace it, by reducing the range of parameter values we need to search. In Fig. 5.6, we show the before (blue) and after (orange) distributions for translation and rotation. By using the predictor, we gain the most from rotation, where prior to alignment we would need to search from $-20°$ to $20°$, whereas after alignment we could get away with $-10°$ to $10°$ or even $-8°$ to $8°$. We see gains for x and y translations as well, but these are less significant.



(a) x-translation (b) y-translation (c) Rotation

Figure 5.6: Distribution of translation and rotation parameters needed to transform the input pose to the ground-truth pose before and after applying the transformation from the alignment predictor. (Best viewed in color.)

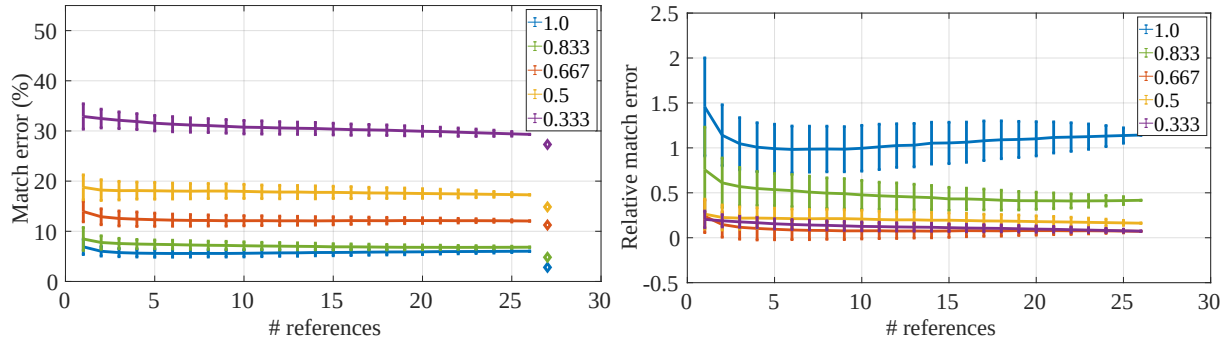


Figure 5.7: Comparing top-1 match error of query shoes of different sized crops. The query is searched against the database after being aligned to a fixed number of references chosen at random from the reference set—if six references are sampled then six versions of the query is searched against the database. The left panel shows the raw match error, where markers represent the error when using the ground-truth alignment. The right panel shows the relative error between prediction and ground-truth.

5.4.2 Matching

We also evaluate how predicting the alignment compares to using the ground-truth alignment for shoe retrieval. The matching pipeline mirrors that of Chapter 4, where we use a Siamese network to perform matching. We use a pre-trained ResNet-34 model to extract the 512-channel activations 'res5cx' for comparison, and compute the similarity of pairs using the multi-channel normalized cross-correlation metric without any learned parameters. In our experiments, we define the full set of references to be the 26 median images of each class—as images within each class are well aligned this equates to matching against the class with the optimal alignment.

In our first experiment, we sample varying size subsets of references to see how sensitive matching performance (top-1 error) is the alignment for different size patches. To be clear, the size of the subset indicates the number of times a query is searched against the database, one for each reference alignment; we take the max over the set of similarity scores for each query-database-item pair as the final score for the pair. The left panel of Fig. 5.7 shows that predicting alignment performs quite well compared to using the ground-truth alignment. For all patch sizes, we see the error reduces as we use more reference alignments—indicating

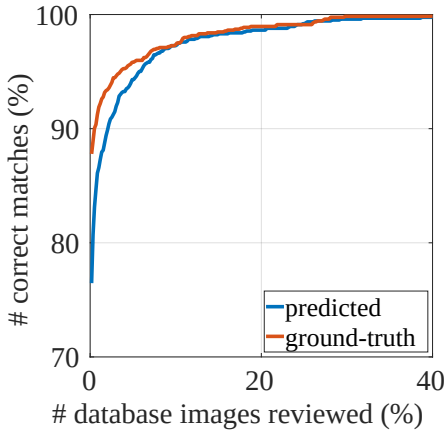


Figure 5.8: Comparing ground-truth alignment and predicted alignment for retrieving relevant shoe images using cumulative match characteristic (CMC).

misalignment across shoe classes. More reference alignments are generally better except for the case of patch size equal to 1.0, where we see slight increase in error. This is emphasized in the right panel of Fig. 5.7 where we show the relative error of prediction to explicit search. Overall this indicates alignment prediction is a viable replacement for explicit search.

In our second experiment, we evaluate how predicted alignment affects retrieval performance (cumulative match characteristic). Unlike the last experiment where we searched the entire database with each reference alignment, in this experiment each reference alignment only matches against database items that are from the same class as the reference; this eliminates chance alignment as the query is scored against each database item only once. We see from Fig. 5.8 that errors in the alignment prediction result in an 11% drop in retrieval performance when reviewing only a few database items—with the gap between predicted and ground-truth not closing until 10% of the database has been reviewed. Overall this means that instead of replacing alignment search, pre-aligning the query to reduce the alignment search space may be more attractive when maximizing retrieval performance.

5.5 Conclusion

In this work, we proposed using a spatial transformer network to predict an affine transformation between a pair of shoe images. We outline a procedure that uses shapes along with this alignment prediction model to align images within a set. Results from following the procedure are general positive—the sets of images are generally well aligned—but there are still some failure cases. Upon inspection we find these failures are caused by out-of-plane rotations of the shoe—a transformation that is outside the set of allowable transformations we explored in this work and will be an area we plan to explore in the future. We demonstrate that while alignment prediction for same-domain shoe retrieval is generally positive. When retrieval accuracy is of utmost importance, replacing explicit search with an alignment predictor seems inadequate; instead, pre-aligning the query with prediction to reduce the search space is a viable solution to speed up matching over solely doing explicit search.

Bibliography

[1] A. Babenko and V. Lempitsky. Efficient indexing of billion-scale datasets of deep descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2055–2063, 2016.

[2] W. J. Bodziak. *Footwear impression evidence: detection, recovery and examination*. CRC Press, 1999.

[3] H. Bristow, A. Eriksson, and S. Lucey. Fast convolutional sparse coding. In *Computer Vision and Pattern Recognition (CVPR)*, 2013.

[4] C. Cao, X. Liu, Y. Yang, Y. Yu, J. Wang, Z. Wang, Y. Huang, L. Wang, C. Huang, W. Xu, et al. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. In *International Conference on Computer Vision (ICCV)*, 2015.

[5] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu. Sketch2photo: Internet image montage. In *ACM Transactions on Graphics (TOG)*, volume 28, page 124. ACM, 2009.

[6] A. Coates and A. Y. Ng. Learning feature representations with k-means. In *Neural networks: Tricks of the trade*, pages 561–580. Springer, 2012.

[7] D. Costea and M. Leordeanu. Aerial image geolocalization from recognition and matching of roads and intersections. *arXiv preprint arXiv:1605.08323*, 2016.

[8] F. Dardi, F. Cervelli, and S. Carrato. A texture based shoe retrieval system for shoe marks of real crime scenes. *Image Analysis and Processing–ICIAP 2009*, pages 384–393, 2009.

[9] P. De Chazal, J. Flynn, and R. B. Reilly. Automated processing of shoeprint images based on the fourier transform for use in forensic science. *IEEE transactions on pattern analysis and machine intelligence*, 27(3):341–350, 2005.

[10] M. Divecha and S. Newsam. Large-scale geolocalization of overhead imagery. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 32. ACM, 2016.

[11] D. L. Donoho. Compressed sensing. *IEEE Transactions on information theory*, 2006.

[12] D. L. Donoho. For most large underdetermined systems of linear equations the minimal $\ell 1$-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(6):797–829, 2006.

[13] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image processing*, 2006.

[14] R. B. Fisher and P. Oliver. Multi-variate cross-correlation and image matching. In *Proc. British Machine Vision Conference (BMVC)*, 1995.

[15] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[16] S. Geiss, J. Einax, and K. Danzer. Multivariate correlation analysis and its application in environmental analysis. *Analytica chimica acta*, 242:5–9, 1991.

[17] F. Girosi. An equivalence between sparse approximation and support vector machines. *Neural computation*, 10(6):1455–1480, 1998.

[18] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio. Maxout networks. In *International conference on Machine learning (ICML)*, 2013.

[19] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *International Conference on Machine Learning (ICML)*, 2010.

[20] M. Gueham, A. Bouridane, and D. Crookes. Automatic recognition of partial shoeprints using a correlation filter classifier. In *Machine Vision and Image Processing Conference, 2008. IMVIP'08. International*, pages 37–42. IEEE, 2008.

[21] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. *Computer Vision–ECCV 2012*, pages 459–472, 2012.

[22] F. Heide, W. Heidrich, and G. Wetzstein. Fast and flexible convolutional sparse coding. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[23] G. B. Huang, V. Jain, and E. Learned-Miller. Unsupervised joint alignment of complex images. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[24] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[25] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pages 2017–2025, 2015.

[26] Z. Ji, W. Huang, G. Kenyon, and L. Bettencourt. Hierarchical discriminative sparse coding via bidirectional connections. In *International Joint Converence on Neural Networks (IJCNN)*, 2011.

[27] Z. Jiang, Z. Lin, and L. S. Davis. Learning a discriminative dictionary for sparse coding via label consistent K-SVD. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.

[28] J. Johnson, M. Douze, and H. Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

[29] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. L. Cun. Learning convolutional feature hierarchies for visual recognition. In *Advances in neural information processing systems (NIPS)*, 2010.

[30] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[31] A. Klein, J. Andersson, B. A. Ardekani, J. Ashburner, B. Avants, M.-C. Chiang, G. E. Christensen, D. L. Collins, J. Gee, P. Hellier, et al. Evaluation of 14 nonlinear deformation algorithms applied to human brain mri registration. *Neuroimage*, 46(3):786–802, 2009.

[32] B. Kong and C. C. Fowlkes. Energy-based spherical sparse coding. *CoRR*, abs/1710.01820, 2017.

[33] B. Kong, J. S. Supancic, D. Ramanan, and C. C. Fowlkes. Cross-domain forensic shoeprint matching. In *British Machine Vision Conference (BMVC)*, 2017.

[34] A. Kortylewski. *Model-based image analysis for forensic shoe print recognition*. PhD thesis, University_of_Basel, 2017.

[35] A. Kortylewski, T. Albrecht, and T. Vetter. Unsupervised footwear impression analysis and retrieval from crime scene data. In *Asian Conference on Computer Vision*, pages 644–658. Springer, 2014.

[36] A. Kortylewski and T. Vetter. Probabilistic compositional active basis models for robust pattern recognition. In *British Machine Vision Conference*, 2016.

[37] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[38] H. Larochelle and Y. Bengio. Classification using discriminative restricted boltzmann machines. In *International conference on Machine learning (ICML)*, 2008.

[39] E. G. Learned-Miller. Data driven image models through continuous joint alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):236–250, 2006.

[40] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. A tutorial on energy-based learning. *Predicting structured data*, 2006.

[41] H. C. Lee, R. Ramotowski, and R. Gaensslen. *Advances in fingerprint technology*. CRC press, 2001.

[42] F.-F. Li, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4):594–611, 2006.

[43] X. Li and Y. Guo. Bi-directional representation learning for multi-label classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML KDD)*. 2014.

[44] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 89–96. IEEE, 2011.

[45] K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate analysis (probability and mathematical statistics)*. Academic Press London, 1980.

[46] N. Martin and H. Maes. *Multivariate analysis*. Academic press, 1979.

[47] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 1997.

[48] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *BMVC*, volume 1, page 6, 2015.

[49] P. M. Patil and J. V. Kulkarni. Rotation and intensity invariant shoeprint matching using gabor transform with application to forensic science. *Pattern Recognition*, 42(7):1308–1317, 2009.

[50] M. Pavlou and N. Allinson. Automatic extraction and classification of footwear patterns. *Intelligent Data Engineering and Automated Learning–IDEAL 2006*, pages 721–728, 2006.

[51] J. Popper Shaffer and M. W. Gillo. A multivariate extension of the correlation ratio. *Educational and Psychological Measurement*, 34(3):521–524, 1974.

[52] R. Š. Radim Tyleček. Spatial pattern templates for recognition of objects with regular structure. In *Proc. GCPR*, Saarbrucken, Germany, 2013.

[53] X. Ren and D. Ramanan. Histograms of sparse codes for object detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2013.

[54] N. Richetelli, M. C. Lee, C. A. Lasky, M. E. Gump, and J. A. Speir. Classification of footwear outsole patterns using fourier transform and local interest points. *Forensic science international*, 275:102–109, 2017.

[55] C. J. Rozell, D. H. Johnson, R. G. Baraniuk, and B. A. Olshausen. Sparse coding via thresholding and local competition in neural circuits. *Neural computation*, 2008.

[56] B. C. Russell, J. Sivic, J. Ponce, and H. Dessales. Automatic alignment of paintings and photographs depicting a 3d scene. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 545–552. IEEE, 2011.

[57] T. Senlet, T. El-Gaaly, and A. Elgammal. Hierarchical semantic hashing: Visual localization from buildings on maps. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 2990–2995. IEEE, 2014.

[58] W. Shang, K. Sohn, D. Almeida, and H. Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *International conference on Machine learning (ICML)*, 2016.

[59] 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*, pages 806–813, 2014.

[60] A. Shrivastava, T. Malisiewicz, A. Gupta, and A. A. Efros. Data-driven visual similarity for cross-domain image matching. *ACM Transactions on Graphics (ToG)*, 30(6):154, 2011.

[61] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *International conference on Learning Representations (ICLR) (workshop track)*, 2015.

[62] Y. Tang, S. N. Srihari, H. Kasiviswanathan, and J. J. Corso. Footwear print retrieval system for real crime scene marks. In *International Workshop on Computational Forensics*, pages 88–100. Springer, 2010.

[63] M. Tygert, A. Szlam, S. Chintala, M. Ranzato, Y. Tian, and W. Zaremba. Convolutional networks and learning invariant to homogeneous multiplicative scalings. *arXiv preprint arXiv:1506.08230*, 2015.

[64] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *ACM International Conference on Multimedia*, 2015.

[65] C.-H. Wei and C.-Y. Gwo. Alignment of core point for shoeprint analysis and retrieval. In *Information Science, Electronics and Electrical Engineering (ISEEE), 2014 International Conference on*, volume 2, pages 1069–1072. IEEE, 2014.

[66] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE transactions on pattern analysis and machine intelligence (TPAMI)*, 2009.

[67] T. Xiao, H. Li, W. Ouyang, and X. Wang. Learning deep feature representations with domain guided dropout for person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1249–1258, 2016.

[68] A. Yang, Z. Zhou, A. Balasubramanian, S. Sastry, and Y. Ma. Fast ℓ_1-minimization algorithms for robust face recognition. *Image Processing, IEEE Transactions on*, 2013.

[69] J. Yang, K. Yu, and T. Huang. Supervised translation-invariant sparse coding. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[70] J. Yang and Y. Zhang. Alternating direction algorithms for ℓ_1-problems in compressive sensing. *SIAM Journal on Scientific Computing*, 2011.

[71] Y. Yekutieli, Y. Shor, S. Wiesner, and T. Tsach. Expert assisting computerized system for evaluating the degree of certainty in 2d shoeprints. Technical report, Technical Report, TP-3211, National Institute of Justice, 2012.

[72] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4353–4361, 2015.

[73] J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1592–1599, 2015.

[74] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[75] L. Zhang and N. Allinson. Automatic shoeprint retrieval system for use in forensic investigations. In *UK Workshop On Computational Intelligence*, 2005.

[76] Y. Zhang, Z. Jiang, and L. S. Davis. Discriminative tensor sparse coding for image classification. In *British Machine Vision Conference (BMVC)*, 2013.

[77] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890, 2017.

[78] N. Zhou, Y. Shen, J. Peng, and J. Fan. Learning inter-related visual dictionary for object recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

Appendices

A Convolutional Sparse Coding

In this section, we discuss implementation details for convolutional sparse sparse coding.

A.1 Least-Squares Filters with Constrained Spatial Support

To see why Eq. 2.11 is equivalent to the previous argmin (Eq. 2.10), consider the following problem

$$\begin{aligned}
\mathbf{s}^* &= \arg\min_{\mathbf{s}} \ \|\mathbf{d}-\mathbf{s}\|_2^2 + \boldsymbol{\lambda}^\intercal(\mathbf{d}-\mathbf{s}) \\
&\quad \text{s.t. } \mathbf{s} = \mathbf{C}\mathbf{s} \\
\mathbf{s}^* = \begin{bmatrix} \mathbf{s}_1^* \\ \mathbf{s}_0^* \end{bmatrix} &= \arg\min_{\mathbf{s}_1,\mathbf{s}_0} \ \|\mathbf{d}_1-\mathbf{s}_1\|_2^2 + \|\mathbf{d}_0-\mathbf{s}_0\|_2^2 + \boldsymbol{\lambda}_1^\intercal(\mathbf{d}_1-\mathbf{s}_1) + \boldsymbol{\lambda}_0^\intercal(\mathbf{d}_0-\mathbf{s}_0) \\
&\quad \text{s.t. } \mathbf{s}_0 = \mathbf{0},
\end{aligned}$$

where we split each vector into two sub-vectors: the part that corresponds to small spatial support (denoted with $_1$) and the part that does not (denoted with $_0$). By splitting $\mathbf{s}$ into $\mathbf{s}_1$ and $\mathbf{s}_0$, we can think about each as separate problems as there are no terms that contain

both.

$$\begin{aligned}\mathbf{s}_1^* &= \arg\min_{\mathbf{s}_1} \|\mathbf{d}_1 - \mathbf{s}_1\|_2^2 + \boldsymbol{\lambda}_1^\intercal(\mathbf{d}_1 - \mathbf{s}_1) \\ &= \arg\min_{\mathbf{s}_1} \|\mathbf{T}\mathbf{d} - \mathbf{s}_1\|_2^2 + \mathbf{T}\boldsymbol{\lambda}^\intercal(\mathbf{T}\mathbf{d} - \mathbf{s}_1)\end{aligned} \qquad \begin{aligned}\mathbf{s}_0^* &= \arg\min_{\mathbf{s}_0} \|\mathbf{d}_0 - \mathbf{s}_0\|_2^2 + \boldsymbol{\lambda}_0^\intercal(\mathbf{d}_0 - \mathbf{s}_0) \\ &\text{s.t. } \mathbf{s}_0 = \mathbf{0} \\ &= \arg\min_{\mathbf{s}_0} \|\mathbf{s}_0\|_2^2\end{aligned}$$

So the $\mathbf{s}_1$ problem amounts to an unconstrained linear least squares; the $\mathbf{s}_0$ problem can be transformed into an equivalent unconstrained problem with a trivial solution when certain conditions hold. These conditions are that $\mathbf{d}_0$ and $\boldsymbol{\lambda}_0$ are null vectors. The matrix $\mathbf{C}$ does exactly this, as it preserves the coefficients of any vector corresponding to the small spatial support and zeros out the rest.

A.2 Implementation Details for Convolutional Sparse Coding

During our discussion of subproblems $\mathbf{z}$ (Sec. 2.2.2) and $\mathbf{d}$ (Sec. 2.2.2), we mentioned an efficient variable reordering to find the optimal solution by solving D independent $K \times K$ linear systems. First we define the following matrices

$$\hat{\mathbf{Z}} = \left[\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_K\right]^\intercal, \hat{\mathbf{T}} = \left[\hat{\mathbf{t}}_1, \ldots, \hat{\mathbf{t}}_K\right]^\intercal, \hat{\mathbf{D}} = \left[\hat{\mathbf{d}}_1, \ldots, \hat{\mathbf{d}}_K\right]^\intercal, \hat{\mathbf{S}} = \left[\hat{\mathbf{s}}_1, \ldots, \hat{\mathbf{s}}_K\right]^\intercal$$
$$\hat{\boldsymbol{\Lambda}}_{\mathbf{t}} = \left[\hat{\boldsymbol{\lambda}}_{\mathbf{t}1}, \ldots, \hat{\boldsymbol{\lambda}}_{\mathbf{t}K}\right]^\intercal, \hat{\boldsymbol{\Lambda}}_{\mathbf{s}} = \left[\hat{\boldsymbol{\lambda}}_{\mathbf{s}1}, \ldots, \hat{\boldsymbol{\lambda}}_{\mathbf{s}K}\right]^\intercal$$
$$\hat{\mathbf{D}\mathbf{x}} = \left[\overline{\hat{\mathbf{d}}_1} \odot \hat{\mathbf{x}}, \ldots, \overline{\hat{\mathbf{d}}_K} \odot \hat{\mathbf{x}}\right]^\intercal, \hat{\mathbf{Z}\mathbf{x}} = \left[\overline{\hat{\mathbf{z}}_1} \odot \hat{\mathbf{x}}, \ldots, \overline{\hat{\mathbf{z}}_K} \odot \hat{\mathbf{x}}\right]^\intercal$$

and denote $\hat{\mathbf{Z}}_i, \hat{\mathbf{T}}_i, \hat{\mathbf{D}}_i, \hat{\mathbf{S}}_i, \hat{\boldsymbol{\Lambda}}_{\mathbf{t}i}, \hat{\boldsymbol{\Lambda}}_{\mathbf{s}i}, \hat{\mathbf{D}\mathbf{x}}_i$, and $\hat{\mathbf{Z}\mathbf{x}}_i$ to be the i-th column of their respective matrices.[1] $\overline{\hat{\mathbf{d}}}$ and $\overline{\hat{\mathbf{z}}}$ denote the conjugate of $\hat{\mathbf{d}}$ and $\hat{\mathbf{z}}$ respectively. We can then solve the

[1]Note that $\hat{\mathbf{T}}$ is unrelated to the truncation matrix $\mathbf{T}$

subproblem $\mathbf{z}$ with

$$\hat{\mathbf{Z}}_i^* = (\hat{\mathbf{D}}_i\hat{\mathbf{D}}_i^\dagger + \mu_\mathbf{t}\mathbf{I})^{-1}(\hat{\mathbf{D}}\mathbf{x}_i + \mu_\mathbf{t}\hat{\mathbf{T}}_i + \hat{\mathbf{\Lambda}}_{\mathbf{t}i}),$$

and subproblem $\mathbf{d}$ with

$$\hat{\mathbf{D}}_i^* = (\hat{\mathbf{Z}}_i\hat{\mathbf{Z}}_i^\dagger + \mu_\mathbf{s}\mathbf{I})^{-1}(\hat{\mathbf{Z}}\mathbf{x}_i + \mu_\mathbf{s}\hat{\mathbf{S}}_i + \hat{\mathbf{\Lambda}}_{\mathbf{s}i}).$$

In Eq. 2.9, we solved for $\hat{\mathbf{d}}_k$ which the inverse Fourier transform needs to be applied to get $\mathbf{d}_k$. However, since only a sub-vector of $\mathbf{d}_k$ is needed computing the inverse of the entire vector is wasteful. If we look at the definition of $\mathbf{C}$, we see that we can define a new matrix $\mathbf{\Phi} = \mathbf{T}\mathbf{F}^{-1}$ that will invert just the part of the vector we need. And the smaller M is with respect to D, the greater the savings we gain. We can redefine Eq. 2.12 and Eq. 2.13 as

$$\tilde{\mathbf{s}}_k = \mathbf{\Phi}\hat{\mathbf{d}}_k + \frac{1}{\mu_\mathbf{s}}\mathbf{\Phi}\hat{\boldsymbol{\lambda}}_{\mathbf{s}k}. \tag{A.1}$$

$$\mathbf{s}_k^* = \mathbf{P}\begin{cases} \|\tilde{\mathbf{s}}_k\|_2^{-1}\tilde{\mathbf{s}}_k & \text{if } \|\tilde{\mathbf{s}}_k\|_2^2 \geq 1 \\ \tilde{\mathbf{s}}_k & \text{otherwise} \end{cases} \tag{A.2}$$

Our entire discussion thus far has revolved around a single image, to handle multiple images is quite straightforward and only requires us to change Eq. 2.9,

$$\hat{\mathbf{d}}^* = (\sum_{x=1}^{N}\hat{\mathbf{Z}}_x^\dagger\hat{\mathbf{Z}}_x + \mu_\mathbf{s}\mathbf{I})^{-1}(\sum_{x=1}^{N}\hat{\mathbf{Z}}_x^\dagger\hat{\mathbf{x}}_x + \mu_\mathbf{s}\hat{\mathbf{s}} - \hat{\boldsymbol{\lambda}}_\mathbf{s}). \tag{A.3}$$

B Energy-Based Spherical Sparse Coding

In this section, we provide additional details on the equivalence of spherical sparse coding to convolutional sparse coding and a proof for the solution for the energy-based spherical sparse coding problem.

B.1 Equivalence of Spherical Sparse Coding and Convolutional Sparse Coding

Here we show that spherical sparse coding (SSC) with a norm constraint on the reconstruction is equivalent to standard convolutional sparse coding (CSC). Expanding the least squares reconstruction error and dropping the constant term $\|x\|_2^2$ gives the CSC problem:

$$\max_{\mathbf{z}} \; 2\mathbf{x}^\intercal\big(\sum_{k=1}^{K}\mathbf{d}_k * \mathbf{z}_k\big) - \|\sum_{k=1}^{K}\mathbf{d}_k * \mathbf{z}_k\|_2^2 - \beta\sum_{k=1}^{K}\|\mathbf{z}_k\|_1.$$

Let $\epsilon = \|\sum_{k=1}^{K}\mathbf{d}_k * \mathbf{z}_k\|_2$ be the norm of the reconstruction for some code $\mathbf{z}$ and let $\mathbf{u}$ be the reconstruction scaled ϵ to have unit norm so that:

$$\mathbf{u} = \frac{\sum_{k=1}^{K}\mathbf{d}_k * \mathbf{z}_k}{\|\sum_{k=1}^{K}\mathbf{d}_k * \mathbf{z}_k\|_2} = \sum_{k=1}^{K}\mathbf{d}_k * \bar{\mathbf{z}}_k \quad \text{with} \quad \bar{\mathbf{z}} = \frac{1}{\epsilon}\mathbf{z}$$

We rewrite the least-squares objective in terms of these new variables:

$$\begin{aligned}\max_{\bar{\mathbf{z}},\epsilon>0} g(\bar{\mathbf{z}},\epsilon) &= \max_{\bar{\mathbf{z}},\epsilon>0} 2\mathbf{x}^\intercal\big(\epsilon\mathbf{u}\big) - \|\epsilon\mathbf{u}\|_2^2 - \beta\|\epsilon\bar{\mathbf{z}}\|_1 \\ &= \max_{\bar{\mathbf{z}},\epsilon>0} 2\epsilon\big(\mathbf{x}^\intercal\mathbf{u} - \frac{\beta}{2}\|\bar{\mathbf{z}}\|_1\big) - \epsilon^2\end{aligned}$$

Taking the derivative of g w.r.t., ϵ yields the optimal scaling ϵ^* as a function of $\bar{\mathbf{z}}$:

$$\epsilon(\bar{\mathbf{z}})^* = \mathbf{x}^\intercal\mathbf{u} - \frac{\beta}{2}\|\bar{\mathbf{z}}\|_1.$$

Plugging $\epsilon(\bar{\mathbf{z}})^*$ back into g yields:

$$\max_{\bar{\mathbf{z}},\epsilon>0} g(\bar{\mathbf{z}},\epsilon) = \max_{\bar{\mathbf{z}},\|u\|_2=1} \big(\mathbf{x}^\intercal \mathbf{u} - \frac{\beta}{2}\|\bar{\mathbf{z}}\|_1\big)^2.$$

Discarding solutions with $\epsilon < 0$ can be achieved by simply dropping the square which results in the final constrained problem:

$$\begin{aligned} &\arg\max_{\bar{\mathbf{z}}} \mathbf{x}^\intercal\big(\sum_{k=1}^{K} \mathbf{d}_k * \bar{\mathbf{z}}_k\big) - \frac{\beta}{2}\sum_{k=1}^{K}\|\bar{\mathbf{z}}_k\|_1 \\ &\quad \text{s.t.} \quad \|\sum_{k=1}^{K} \mathbf{d}_k * \bar{\mathbf{z}}_k\|_2 \leq 1. \end{aligned}$$

B.2 Energy-Based Spherical Sparse Coding Solution Proof

We show in this section that coding in the Energy-Based Spherical Sparse Coding (EB-SSC) model can be solved efficiently by a combination of convolution, shrinkage and projection, steps which can be implemented with standard libraries on a GPU. For convenience, we first rewrite the objective in terms of cross-correlation rather than convolution (*i.e.*, $\mathbf{x}^\intercal(\mathbf{d}_k * \mathbf{z}_k) = (\mathbf{d}_k \star \mathbf{x})^\intercal \mathbf{z}_k$). For ease of understanding, we first consider the coding problem when there is no classification term.

$$\mathbf{z}^* = \arg\max_{\|\mathbf{z}\|_2^2 \leq 1} \mathbf{v}^\intercal \mathbf{z} - \beta\|\mathbf{z}\|_1,$$

where $\mathbf{v} = [(\mathbf{d}_1 \star \mathbf{x})^\intercal, \ldots, (\mathbf{d}_K \star \mathbf{x})^\intercal]^\intercal$. Pulling the constraint into the objective, we get its Lagrangian function:

$$\mathcal{L}(\mathbf{z},\lambda) = \mathbf{v}^\intercal \mathbf{z} - \beta\|\mathbf{z}\|_1 + \lambda\big(1 - \|\mathbf{z}\|_2^2\big).$$

From the partial subderivative of the Lagrangian w.r.t., z_i we derive the optimal solution as a function of λ; and from that find the conditions in which the solutions hold, giving us:

$$z_i(\lambda)^* = \frac{1}{2\lambda} \cdot \begin{cases} v_i - \beta & v_i > \beta \\ 0 & \text{otherwise} \\ v_i + \beta & v_i < \beta \end{cases} . \tag{B.4}$$

This can also be compactly written as:

$$\begin{aligned} \mathbf{z}(\lambda)^* &= \frac{1}{2\lambda}\tilde{\mathbf{z}}, \\ \tilde{\mathbf{z}} &= \mathbf{s}^2 \odot \mathbf{v} - \beta\mathbf{s} \end{aligned} \tag{B.5}$$

where $\mathbf{s} = \text{sign}(\mathbf{z}^*) \in \{-1, 0, 1\}^{|\mathbf{z}|}$ and $\mathbf{s}^2 = \mathbf{s} \odot \mathbf{s} \in \{0, 1\}^{|\mathbf{z}|}$. The sign vector of $\mathbf{z}^*$ can be determined without knowing λ, as λ is a Lagrangian multiplier for an inequality it must be non-negative and therefore does not change the sign of the optimal solution. Lastly, we define the squared ℓ_2-norm of $\tilde{\mathbf{z}}$, a result that will be used later:

$$\begin{aligned} \|\tilde{\mathbf{z}}\|_2^2 &= \tilde{\mathbf{z}}^\intercal(\mathbf{s}^2 \odot \mathbf{v}) - \beta\tilde{\mathbf{z}}^\intercal\mathbf{s} \\ &= \tilde{\mathbf{z}}^\intercal\mathbf{v} - \beta\|\tilde{\mathbf{z}}\|_1. \end{aligned} \tag{B.6}$$

Substituting $\mathbf{z}(\lambda)^*$ back into the Lagrangian we get:

$$\mathcal{L}(\mathbf{z}(\lambda)^*, \lambda) = \frac{1}{2\lambda}\mathbf{v}^\intercal\tilde{\mathbf{z}} - \frac{\beta}{2\lambda}\|\tilde{\mathbf{z}}\|_1 + \lambda\big(1 - \frac{1}{4\lambda^2}\|\tilde{\mathbf{z}}\|_2^2\big),$$

and the derivative w.r.t., λ is:

$$\frac{\partial\mathcal{L}(\mathbf{z}(\lambda)^*}{\partial\lambda} = -\frac{1}{2\lambda^2}\mathbf{v}^\intercal\tilde{\mathbf{z}} + \frac{\beta}{2\lambda^2}\|\tilde{\mathbf{z}}\|_1 + 1 + \frac{1}{4\lambda^2}\|\tilde{\mathbf{z}}\|_2^2.$$

Setting the derivative equal to zero and using the result from Eq. B.6, we can find the optimal solution to λ:

$$\begin{aligned} \lambda^2 &= \frac{1}{2}\tilde{\mathbf{z}}^\intercal \mathbf{v} - \frac{\beta}{2}\|\tilde{\mathbf{z}}\|_1 - \frac{1}{4}\|\tilde{\mathbf{z}}\|_2^2 = \frac{1}{2}\|\tilde{\mathbf{z}}\|_2^2 - \frac{1}{4}\|\tilde{\mathbf{z}}\|_2^2 \\ \implies \lambda^* &= \frac{1}{2}\|\tilde{\mathbf{z}}\|_2. \end{aligned}$$

Finally, plugging λ^* into Eq. B.5 we find the optimal solution

$$\mathbf{z}^* = \frac{\tilde{\mathbf{z}}}{\|\tilde{\mathbf{z}}\|_2}. \tag{B.7}$$