Visual Scene Understanding by Deep Fisher Discriminant Learning

Arash Shahriari

A thesis submitted for the degree of

DOCTOR of PHILOSOPHY

The Australian National University

October 2017

I declare that except where due reference is provided,
the work presented in this thesis is my own original research.

Arash Shahriari
23 October 2017

I dedicate this thesis to my wife

PARISA

a lovely angel from the heaven

for all of her understanding, support and patience.

Acknowledgements

I hereby acknowledge the Australian National University (ANU), National Information and Communication Technology Australia (NICTA), and Commonwealth Scientific and Industrial Research Organisation (CSIRO), for providing me with a great opportunity to undertake the research for my PhD studies. This research is supported by an Australian Research Training Program (RTP) Scholarship. I express my very sincere gratitude to my supervisory chair; Professor Henry Gardner for his all-out efforts to bring back an excellent supervision to my PhD program. I extend a special thanks to my supervisory panel: Professor Fatih Porikli, Associate Professor Stephen Gould, and Associate Professor Hongdong Li for all their valuable advice and kind support. I am grateful to Dr. Mehrtash Harandi, Dr. Nariman Habili, and Dr. Cong Phuoc Huynh for fruitful debates and their helpful comments. I would also acknowledge Dr. Antonio Robles-Kelly and Dr. Jose M. Alvarez for their assistance during the early stages of this project. Proofreading of this thesis was performed by A+ Academic Editing Canberra (Kathie Brown).

Abstract

Modern deep learning has recently revolutionized several fields of classic machine learning and computer vision, such as, scene understanding, natural language processing and machine translation. The substitution of feature hand-crafting with automatic feature learning, provides an excellent opportunity for gaining an in-depth understanding of large-scale data statistics. Deep neural networks generally train models with huge numbers of parameters, facilitating efficient search for optimal and sub-optimal spaces of highly non-convex objective functions. On the other hand, Fisher discriminant analysis has been widely employed to impose class discrepancy, for the sake of segmentation, classification, and recognition tasks. This thesis bridges between contemporary deep learning and classic discriminant analysis, to accommodate some important challenges in visual scene understanding, i.e. semantic segmentation, texture classification, and object recognition. The aim is to accomplish specific tasks in some new high-dimensional spaces, covered by the statistical information of the datasets under study. Inspired by a new formulation of Fisher discriminant analysis, this thesis introduces some novel arrangements of well-known deep learning architectures, to achieve better performances on the targeted missions. The theoretical justifications are based upon a large body of experimental work, and consolidate the contribution of the proposed idea; Deep Fisher Discriminant Learning, to several challenges in visual scene understanding.

Contents

List of Figures

List of Tables

List of Abbreviations

AEN Autoencoder Network.

BM Boltzmann Machines.

BPA Basic Probability Assignment.

CCA Canonical Correlations Analysis.

CDBN Convolutional Deep Belief Network.

CNN Convolutional Neural Network.

CoNN Condensed Nearest Neighbour.

CRBM Convolutional Restricted Boltzmann Machines.

DBN Deep Belief Network (Stacked Boltzmann Machines).

DCIGN Deep Convolutional Inverse Graphics Network.

DFK Deep Fisher Kernel.

DFN Deep Fisher Network.

DGDA Deep Generalized Discriminant Analysis.

DLDA Deep Linear Discriminant Analysis.

DN Deconvolutional Network.

DRN Deep Residual Network.

ELM Extreme Learning Machines.

ENN Edited Nearest Neighbour.

ENNth	Edited Nearest Neighbour Estimating Class Probabilistic and Threshold.
ENRBF	Edited Normalized Radial Basis Function.
ESN	Echo State Network.
FA	Factor Analysis.
FDA	Fisher Discriminant Analysis.
FISTA	Fast Iterative Shrinkage Thresholding Algorithm.
FNN	Feedforward Neural Network.
FNNC	Fast Nearest Neighbour Condensation.
FVN	Fisher Vector Network.
GAN	Generative Adversarial Network.
GRU	Gated Recurrent Unit.
HDA	Heteroscedastic Discriminant Analysis.
HFV	Hybrid Fisher Vectors.
HNN	Hopfield Neural network.
KN	Kohonen Network.
KNN	K-Nearest Neighbour.
KPCA	Kernel Principal Component Analysis.
LDA	Linear Discriminant Analysis.
LPP	Locality Preserving Projection.
LR	Linear Regression.
LSM	Liquid State Machines.

LSTM Long-Short Term Memory Network.

MAF Maximum Autocorrelation Factors.

MCS Minimal Consistent Set.

MDS Multi-Dimensional Scaling.

MMC Maximum Margin Criterion.

NCNEdit Nearest Centroid Neighbour Edition.

NDCG Normalized Discounted Cumulative Gain.

NSC Nearest Shrunken Centroids.

NTM Neural Turing Machines.

PCA Principal Component Analysis.

PPCA Probabilistic Principal Component Analysis.

PSC Prototype Selection by Clustering.

RBF Radial Basis Function.

RBM Restricted Boltzmann Machines.

RENN Repeated Edited Nearest Neighbour.

RNN Recurrent Neural Network.

SDR Sufficient Dimensionality Reduction.

SFA Slow Feature Analysis.

SNN Selective Nearest Neighbour.

SVD Singular Value Decomposition.

SVM Support Vector Machines.

UICA Undercomplete Independent Component Analysis.

Chapter 1

Introduction

Fisher Discriminant Analysis (FDA) is a well-known statistical method, used to tackle the curse of dimensionality and to provide efficient supervised classification. Recent advances in the field of artificial neural networks, have led to a successful trend for automatic feature crafting, known as, deep learning. Visual scene understanding as a fundamental challenge in modern computer vision, has recently been addressed with considerably higher precision, using deep learning algorithms. This thesis investigates the unification of FDA and deep learning, to introduce some novel deep neural architectures that deliver better outcomes than classic deep neural networks. The proposed architectures are specified to visual scene understanding tasks, such as semantic segmentation, texture classification, and object recognition. Extensive experiments confirm that, they are able to improve state-of-the-art performances in various standard benchmarks of computer vision.

1.1 Overview

Linear dimensionality reduction has become the cornerstone analysis of high-dimensional data, due to its simple geometrical interpretation and its attractive computational properties. This captures many features of interest in data, such as, covariance, dynamical structure, correlation between datasets, input-output relationships, and the margins between data classes. A variety of methods have been developed for linear dimensionality reduction, one of which is FDA.

1.1.1 Semantic Segmentation

Reformulation of FDA introduces a novel, supervised colour transformation that helps to improve the performance of pixelwise classification. This transformation uses class information to project the initial colour space into an intermediate high-dimensional space, spanned by the number of classes. This then follows with a subspace projection to make a new colour space, with the same dimension as the primary colour space.

The reason is that, any standard colour transformation needs to map the primary colours into the same number of channels and range of colours. To comply with the primary range of colours, anisotropic scaling-translation is applied to the projected colour space. The pipeline of projection, backprojection, and anisotropic transformation, produces a nonlinear mapping of the primary colour space into a target space, with higher class distinction. The target colour space significantly enhances the precision of semantic segmentation by imposing better class separation, learned from the distribution of data and classes.

The projection is inspired by FDA and the backprojection is applied through metric Multi-Dimensional Scaling (MDS). The former tries to improve the distinction across classes by minimizing the ratio of inter-intra class scatterings, while the latter preserves the imposed discrepancy, by means of a metric subspace mapping. They are followed by the anisotropic scaling-translation to form the target colour space. The extensive experiments show that, the supervised colour transformation can produce new colour spaces, that outperform several standard colour spaces in pixelwise semantic segmentation.

1.1.2 Texture Classification

For the purpose of texture understanding, a specific Convolutional Neural Network (CNN) is introduced, that tries to address texture classification by learning the scale, orientation and resolution of each texture filters, rather than the whole filter itself. Using a FDA-inspired objective function, this novel architecture is able to train a large bank of distributed texture filters in several variable-depth layers, and makes an ensemble of convolutional features, for the sake of texture recognition.

Another contribution is a new Autoencoder Network (AEN), that is able to learn separable and parametric texture filters, by a FDA-based coder-decoder formulation. This network is able to train a banks of texture filters and aggregate the convolutional responses, to form an ensemble of high-distinction features, for the texture recognition. The experiments are conducted on several publicly-available texture datasets, varying in the number of classes and the quality of texture samples. The results confirm the advantage of the proposed deep architectures, to enhance the precision of texture classification, in standard benchmarks.

1.1.3 Object Recognition

One of the preliminary tasks in visual scene understanding is object recognition. To address this challenge by deep learning, some novel contributions to pooling, transfer learning and multi-objective learning, are made in this thesis.

A supervised rank pooling method is discussed, that selects the most informative activations by learning from the discrepancy among deep convolutional responses in pooling layers. Here, multipartite ranking is used to score the features, based on the class discriminations (Fisher's criterion) in the dataset under study; this then picks the high-ranked features from each pooling window, instead of the average or maximum of the assignments.

The complex wiring between neural layers and imbalanced distribution of data, make it a hard task to transfer learning between deep convolutional neural networks. Instead of fine-tuning the whole fully-connected layers, a distributed backpropagation is introduced that feeds each convolutional filters separately, whilst backpropagates all the filters jointly, using basic probability assignment of the evidence theory. Consequently, the trained convolutional filters in primary domain, are transferred to the target domain, considering their individual contributions, as well as, their joint gradients.

Deploying of various objective functions in a deep convolutional network, is proved advantageous to improve the overall classification performance. A unified backpropagation scheme is proposed to link the optimization of multi-objective cost functions, including Softmax, Support Vector Machines (SVM), and Linear Discriminant Analysis (LDA). This tackles the extensive burden of boosting for various objective functions or complex formulation of multi-objective gradients, by employing basic probability assignment. In practice, the unified backpropagation links gradients of multi-objective losses to update the parameters.

1.2 Outline

The remainder of this thesis is divided into a series of parts and chapters, where the challenges outlined above, are discussed in detail.

- Chapter 2: Literature Review

  A comprehensive survey of discriminant analysis methods and deep learning architectures, is provided and contributions of this thesis, are contrasted against.

- Chapter 3: Discriminant Analysis

  The classic linear dimensionality reduction is introduced, and then, its formulation is extended to define a supervised projection into high-dimensional spaces, spanned by the number of classes in the dataset at hand.

- Part I (Chapter 4): Semantic Segmentation

  A supervised colour transformation for pixelwise semantic segmentation is presented. This transformation aims to use FDA for imposing the highest possible discrimination between classes, that benefits any classifier of choice.

- Part II (Chapters 5, 6): Texture Classification

  Inspired by FDA, some novel deep architectures; Fisher CNN and Fisher AEN, are introduced to learn banks of parametric or separable filters, for texture understanding.

- Part III (Chapters 7, 8, 9): Object Recognition

  Some contributions towards core concepts of deep learning, targeting object recognition, are explained. They employ FDA for a new pooling scheme, and use basic probability assignment of the evidence theory, to perform transfer learning (distributed backpropagation) and multi-objective learning (unified backpropagation).

- Chapter 10: Conclusion

  All of the above discussions are wrapped up and a summary is provided. Finally, some ideas for future work, are presented.

1.3 Publications

The material presented here, encompasses several peer-reviewed and open access contributions to the filed as follows,

- Shahriari A., "Unified Backpropagation for Multi-Objective Deep Learning," arXiv:1710.07438, October 2017
- Shahriari A., "Distributed Deep Transfer Learning by Basic Probability Assignment," arXiv:1710.07437, October 2017
- Shahriari A., Porikli F., "Multipartite Pooling for Deep Convolutional Neural Networks," arXiv:1710.07435, October 2017
- Shahriari A., "Parametric Learning of Texture Filters by Stacked Fisher Autoencoders," Digital Image Computing: Techniques and Applications (DICTA'16), Gold Coast, Australia, December 2016
- Shahriari A., "Learning of Separable Filters by Stacked Fisher Convolutional Autoencoders," The British Machine Vision Conference (BMVC'16), York, UK, September 2016
- Shahriari A., "Learning Deep Filter Banks in Parallel for Texture Recognition," International Conference on Image Processing (ICIP'16), Phoenix, Arizona, USA, September 2016 (Lecture)
- Shahriari A., "Multipartite Ranking-Selection of Low-Dimensional Instances by Supervised Projection to High-Dimensional Space," arXiv:1606.07575, June 2016
- Shahriari A., Alvarez J.M., Robles-Kelly A., "Class-driven Color Transformation for Semantic Labeling," Asian Conference on Computer Vision (ACCV'14), Singapore, November 2014

Chapter 2

Literature Review

This chapter provides a survey of discriminant analysis and deep learning. The treatment here, avoids complex formulations in favour of presenting a narrative, comprising short descriptions of the fundamental concepts, relevant to this thesis.

2.1 Discriminant Analysis

Linear dimensionality reduction has been used for the analysis of high-dimensional data, and it is a well-known concept of statistical machine learning. This technique computes a linear mapping from the original high-dimensional data into a low-dimensional representation, while preserves specific statistics of the data. It can be used for data visualization, structural analysis, feature extraction, and compression. This survey introduces algorithms of linear dimensionality reduction, followed by discussions on linear discriminant analysis. These are specially formulated from an optimization viewpoint (Cunningham and Ghahramani [2015]).

2.1.1 Linear Dimensionality Reduction

To start, minimizing the sum of squared residual errors between projected and original data points, or maximizing the variance deviation, is the general formulation of Principal Component Analysis (PCA) (Eckart and Young [1936]). The solution corresponds to the largest eigenvalues, produced by singular value decomposition of the covariance matrix. Probabilistic Principal Component Analysis (PPCA) (Theobald [1975]), uses PCA to partition the data into an orthogonal set of noise and a linear mapping into lower dimensions.

The more general case of PCA is Factor Analysis (FA) (Spearman [1904]). This fits the noise per observation rather to all observations, which adds scale-invariance to each measurement, but loses the rotational invariance across observations. When the data has some associated class labels, FDA (Fisher [1936]) projects the original data, in order to maximize the separation between classes. As a result, the data covariance is partitioned into the covariance contributed within each class and the covariance between each class. FDA assumes Gaussian distributions with the same variance.

Taking into account multiple datasets, Canonical Correlations Analysis (CCA) (Hotelling [1936]), tries to find a set of low-dimensional mappings, that maximizes the correlations between projected datasets. Maximum Autocorrelation Factors (MAF) (Switzer and Green [1984]), assumes a predefined order for the high-dimensional data to preserve temporally-interesting structure, and seeks a low-dimensional representation that includes temporal structure. Undercomplete Independent Component Analysis (UICA) (Hyvärinen et al. [2001]), preprocesses mixed data with PCA to reduce the data to lower dimensions, and then, runs a standard independent component analysis.

Slow Feature Analysis (SFA) (Wiskott [2003]), recovers a slowly moving projection of data, that may produce a meaningful representation of the features, when the measured data can have rapidly changing values over time, even though the features of interest move more slowly. This minimizes the trace of projection of the covariance. In contrast to PCA that minimizes low-dimensional reconstruction error, MDS (Borg and Groenen [2005]) maximizes the scatter of projection, to yield the most informative projection. Locality Preserving Projection (LPP) (He et al. [2005]), considers local neighbourhood structure. This avoids outliers and nonlinear distortion in the data, by constructing a neighbourhood graph of the training data, and by using that to define the loss function.

For better feature selection in a supervised learning regime, Sufficient Dimensionality Reduction (SDR) (Johnstone and Titterington [2009]), finds an orthogonal projection of the data. The reduced-dimension points capture all statistical dependency between high-dimensional covariates and responses. One of the most popular methods in statistical modelling is Linear Regression (LR) (Adragni and Cook [2009]), that maps high-dimensional data onto a low-dimensional hyperplane, defined by the number of independent variables.

2.1.2 Linear Discriminant Analysis

LDA is a widely-used supervised dimensionality reduction method in computer vision and pattern recognition. This aims to impose the highest possible discrepancy among classes, by maximizing the between-class distances, whilst minimizing the within-class scattering (Fisher [1936]). Numbers of extensions to the classic LDA, have been proposed in the literature.

Subspace LDA (Zhao et al. [1999]), first employs PCA to project the data from the original vector space into a subspace, where the subspace dimension is carefully chosen, and then uses LDA to obtain a linear classifier in this subspace. This also employs a weighted distance metric, guided by LDA eigenvalues, to improve the performance. Null-space LDA (Chen et al. [2000]), proposes a technique to solve the small sample size problem, considering that the most expressive PCA-driven vectors in the null space of the within-class scatter matrix, are equal to the optimal discriminant vectors, derived in the original space by LDA.

Orthogonal Centroid LDA (Park et al. [2003]), provides a mathematical framework for low-dimensional representation in vector space, using a matrix rank reduction formula. This also introduces new methods for dimension reduction based on the centroids of data clusters, when a priori information on the cluster structure of the data, is at hand. Uncorrelated LDA (Ye [2005]), is a generalized discriminant analysis, based on a new optimization criterion which extends the objective function of LDA, when the scatter matrices are singular. The features in this reduced space, are highly uncorrelated, in contrast to orthogonal LDA (Prasad et al. [2010]), that the discriminant vectors are quite orthogonal to each other.

Regularized LDA (Guo et al. [2007]), introduces a modified version of LDA, that generalizes the idea of the Nearest Shrunken Centroids (NSC) (Tibshirani et al. [2003]). Sparse LDA (Ye et al. [2008]), presents a novel formulation of LDA, that employs multivariate linear regression with L1-norm penalty, controlled by a regularization parameter, for feature extraction and visualization. This is based on the equivalence relationship between LDA and the least-squares method for multiclass classification.

In contrast to the dimension reduction methods, this thesis introduces Fisher discrimination for dimension expansion (Shahriari et al. [2014]), imposing better class distinction into the representations in higher dimensions. When this follows by a proper subspace mapping algorithm, the class separation holds to the best advantage of any supervised classifiers.

2.2 Deep Learning

Artificial neural networks are, connected architectures of computational units called neurons. They are generally organized in an input layer, several hidden layers, and an output layer. The adjacent layers are usually fully-connected, so that, every neuron in a layer is linked to every neuron of the next layer. There is a vast variety of deep learning architectures in the literature.

2.2.1 Architectural Overview

Feeding the information from input to output layers, Feedforward Neural Network (FNN) and Perceptron (Rosenblatt [1958]), are trained by error backpropagation. The parameters are updated by gradient of difference between predicted and actual output. However, they are not popular today. Radial Basis Function network (RBF) (Broomhead and Lowe [1988]) employs the same architecture and is activated by radial basis function.

In contrast to perceptrons, each neuron of Hopfield Neural Network (HNN) (Hopfield [1982]), behaves as an input before, a hidden node during, and an output after training. As a result, this only converges to special patterns, preassigned to these neurons in advance. For Kohonen Network (KN) (Kohonen [1982]), neurons are adjusted to match the input, by dragging along their neighbours.

Boltzmann Machines (BM) (Hinton and Sejnowski [1986]), are similar to Hopfield network, but each neuron is specifically marked as an input, hidden or output node. This is initialized randomly and, is trained by the backpropagation, which gives more binary activation patterns to the neurons, as compared to HNN.

Restricted Boltzmann Machines (RBM) (Smolensky [1986]), are the restricted version of HNN and BM, since they do not directly connect input neurons to other input nodes, or hidden neurons to other hidden units. They are trained by forward-backward passes of the data, rather than passing forward and then, backpropagating.

Stacked Boltzmann machines, also called Deep Belief Networks (DBN) (Bengio et al. [2007]), are typically trained stack by stack; which is known as greedy training, through contrastive divergence or backpropagation. They can be used to either generate new data or classify the existing one.

Convolutional Neural Network (CNN) (LeCun et al. [1998]), proves beneficial for various applications of image classification. This network usually crawls an image by neighbouring patches to feed the convolutional layers. By going deeper, these convolutional layers shrink by powers of two and use max pooling to pick the details.

Reversed convolutional neurons form Deconvolutional Network (DN) (Zeiler et al. [2010]), that is fed by a word and responds by generating a corresponding picture. Here, the pooling layers are replaced with the similar inverse operations, for example, min pooling instead of max pooling and so on. Deep Convolutional Inverse Graphics Network (DCIGN) (Kulkarni et al. [2015]), encodes the input by using a CNN and, decodes it with a DN. This is trained by backpropagation and, is also able to model the complex transformations on an image.

Autoencoder Network (AEN) (Bourlard and Kamp [1988]), is similar to feedforward network and encodes information automatically. The hidden layers are usually smaller than the input-output layers; and the layers are arranged symmetrically. The encoding layers sit before and the decoding ones sit after the middle layers. They are generally trained by gradient descent and backpropagation. Variational AEN (Kingma and Welling [2013]), learns by the probability distribution of input. This uses Bayesian mathematics to rule out the nodes that influence each others.

Acting on temporal data streams, Recurrent Neural Network (RNN) (Elman [1990]), connects through time, so that, neurons are fed by other neurons. Hence, the order of feeding matters here. It is not only useful for sound or video sequences, but also, for completing image or text information. The problem of vanishing gradients over time in RNN, is more difficult to avoid than other deep architectures, which only loose information in depth.

To tackle the vanishing gradients, Long-Short Term Memory Network (LSTM) (Hochreiter and Schmidhuber [1997]), uses gates and memory cells. The gates protect information inside memory cells, by controlling the flow of data into them. The input and output gates decide, how much information is entered from previous neurons or, is exported to the next cell. Forget gates lock the sate of the memory cell, to prevent interference from the current and recent states. This network is successful on complex sequences, although they need more computing resources, in comparison with RNN.

Gated Recurrent Unit (GRU) (Chung et al. [2014]), is a variation of LSTM, in terms of gating and wiring. It has an update gate to tune the flow of data, and a reset gate to forget, which makes them faster and easier to run than LSTM. Neural Turing Machines (NTM) (Graves et al. [2014]), do not code the memory cell into a neuron and combine the power of neural networks with addressable memory banks.

Employing a threshold function instead of sigmoid activation, a neuron in Liquid State Machines (LSM) (Maass et al. [2002]), accumulates on itself, until it reaches a certain threshold and fires to other neurons, creating a spiking pattern. Echo State Network (ESN) (Jaeger and Haas [2004]), is a type of RNN, that set random connections between its neurons. This feeds the input, updates the neurons, and observes the output over time. During training, only the connections between the top layer and hidden units are changed. Extreme Learning Machines (ELM) (Cambria et al. [2013]), are another random connected network, like LSM or ESN, but these are employed like a FNN.

Generative Adversarial Network (GAN) (Goodfellow et al. [2014]), is a twin network; one generates contents and the other judges them. The discriminating network receives input or generative data, and the error is calculated based on its prediction regarding the data source. It can be difficult to strike a balance between prediction and generation, for these twins. Deep Residual Network (DRN) (He et al. [2015]), is a very deep FNN with direct connections of one layer, to another deeper layer, plus the next one. This trains very deep representations without the burden of vanishing gradients.

2.2.2 Convolutional Neural Network

Inspired by the organization of animal visual cortex (Hubel and Wiesel [1962]), CNN has shown promising performance in processing of two-dimensional data, such as, single images or video streams (Krizhevsky et al. [2012]). It is the first truly successful deep learning architecture, trained in hierarchical layers by sparse interaction, parameter sharing and equivariant representation (Goodfellow et al. [2016a]). Some variants to the classic CNN architecture, have been introduced in the literature.

Convolutional Restricted Boltzmann Machines (CRBM) (Desjardins and Bengio [2008]), employ RBM in CNN, by computing convolutions with normal RBM, as the kernel. Although the number of parameters in RBM, depends on the dimension of the input image, the complexity of this architecture, only relates to the number of features to be extracted, and the size of the receptive field. Convolutional Deep Belief Network (CDBN) (Krizhevsky and Hinton [2010]) uses a combination of locally-globally connected units, as well as, a few tricks to reduce the effects of overfitting, to achieve better performance for classification.

FFT-based CNN (Mathieu et al. [2013]), presents an algorithm which accelerates training and inference, by computing convolutions as pointwise products in the Fourier domain. This reuses the same transformed feature map, several times. Mel-filter bank CNN (Sainath et al. [2013]), replaces the filter bank with a layer, that is learned jointly with the rest of network, and minimizes cross-entropy, which is more closely tied to a speech recognition objective.

Recursive CNN (Eigen et al. [2013]), addresses the challenge of appropriate sizing of the network, including number of layers, feature maps, kernel sizes, and etc. This focuses on assessing the independent contributions of these linked variables, using a recursive architecture, whose weights are tied between layers.

2.2.3 Autoencoder Network

AEN (Hinton and Salakhutdinov [2006]), learns low-dimensional codes that work better than PCA, to reduce the dimensionality of data, using gradient descent. It fine-tunes the weights by finding an effective way to initialize them. This is trained to encode the input into a representation, so that, the input can be reconstructed from that representation. Hence, AEN continuously extracts some useful features, during the propagation phase, while filters the useless information, at the same time.

Sparse AEN (Ranzato et al. [2006]), learns sparse overcomplete features. It uses a linear encoder and a linear decoder, preceded by a sparsifying nonlinearity, that turns a code vector into a quasi-binary sparse code vector. Given an input, the optimal code minimizes the distance between the output of the decoder and the input patch while being, as similar as possible, to the encoder output.

Multistage AEN (Jarrett et al. [2009]), combines CNN and AEN using some nonlinearities, including rectification and local contrast normalization. Stacked Convolutional AEN (Masci et al. [2011]), is a novel convolutional architecture for unsupervised feature learning. A stack of convolutional autoencoders forms a CNN, that each of the autoencoders, is trained using conventional gradient descent, without additional regularization terms.

Denoising AEN (Vincent [2011]), is a competitive alternative to RBM, for unsupervised pretraining of each layer in a deep architecture. Its training criterion is equivalent to matching the score of a specific energy-based model to that of a non-parametric Parzen density estimator of the data. This suggests a different way to apply score matching for denoising and, does not compute the second derivatives. Stacked Denoising AEN (Vincent et al. [2010]), stacks layers of denoising autoencoders, which are trained locally to denoise corrupted versions of their inputs. It is also able to learn Gabor-like edge detectors from image patches and, larger stroke detectors from digit images.

Contractive AEN (Rifai et al. [2011]), adds a well-chosen penalty term to the classical reconstruction cost function, corresponding to the Frobenius norm of the Jacobian matrix of the encoder activations, with respect to the input. Inspired by improved performance of imposing sparsity, k-Sparse AEN (Makhzani and Frey [2013]), is an autoencoder network with linear activation function, where only the k highest activities are kept in the hidden layers. This is simple to train and its encoding stage is very fast, making it well-suited to large-scale problems, where conventional sparse coding algorithms cannot be applied.

Separable Deep AEN (Sun et al. [2016]), employs deep autoencoders for accurate modelling of the clean data. Then, extra deep autoencoders are introduced to represent the residual part, obtained by subtracting the estimated clean data from noisy data. The enhanced representation is thus obtained, by transforming this back into the time domain.

2.2.4 Deep Discriminant Analysis

Composition of LDA and deep learning, has been proposed in recent publications. This mostly employs LDA or its extensions, as objective functions on the top of a deep neural network. Here, the contributions of this thesis contrasts against what has already been done.

Deep Generalized Discriminant Analysis (DGDA) (Stuhlsatz et al. [2012]), assumes discriminative features, generated from independent Gaussian class-conditionals. Since LDA employs linear discriminant function, it is insufficient to extract optimal discriminative features from arbitrarily distributed raw measurements. DGDA uses nonlinear transformations, that are learned by deep neural networks in a semi-supervised manner. This maps high-dimensional input data into a low-dimensional representation, to facilitate accurate prediction.

In this thesis, Fisher CNN (Shahriari [2016a]), introduces a new combination of FDA and CNN, that applies Fisher discrimination in a layer-wise architecture, to generate a high-dimensional convolutional representation. In contrast to DGDA, this uses supervised learning to project the low-dimensional input, into higher dimensions, spanned by the number of classes in the dataset under study.

Deep Linear Discriminant Analysis (DLDA) (Dorfer et al. [2015]), learns a nonlinear extension of LDA, that preserves class separability, for the purpose of linear dimensionality reduction and classification. Instead of maximizing the likelihood of target labels for individual samples by categorical cross entropy, this proposes an objective function to produce features with low variance within the same class and high variance between different classes. The objective function of DLDA, is derived from the general LDA eigenvalue problem and, is trained with the classic backpropagation.

The Fisher CNN is also different from DLDA, because the objective function still remains a categorical cross entropy, and FDA is only employed for supervised feature crafting. It is also able to learn the convolutional filters in a distributed manner, while the depth of layers varies for each filter, in accordance with its capability to make better distinction among classes.

Deep Fisher Network (DFN) (Simonyan et al. [2013]), introduces discriminatingly trained convolutional neural networks, by stacking Fisher vector encoding, in multiple layers. This significantly improves the precision over standard Fisher vectors, and obtains competitive results with deep convolutional networks, at a smaller computational cost. Deep Fisher Kernel (DFK) (Sydorov et al. [2014]) unifies Fisher kernels and deep learning to transfer ideas from one domain to the other, by interpreting a multilayer feed-forward network. The final layer is the classifier, parametrized by a weight vector, and the two previous layers compute Fisher vectors, parametrized by the coefficients of a Gaussian mixture model.

This thesis presents Fisher AEN (Shahriari [2016b]), that is a novel way of linking the FDA with AEN. This replaces coder-encoder layers with dimension expansion-reduction modules. It employs Fisher vector encoding to make a high-distinction dictionary at the top layer, that is in contrast to DFN, which stacks Fisher vector encoding, in a multilayer architecture.

Hybrid Fisher Vectors (HFV) (Perronnin and Larlus [2015]), combine the strengths of Fisher vectors and deep learning. The early unsupervised layers rely on the Fisher vectors, while the subsequent fully-connected supervised layers, are trained with backpropagation. Fisher Vector Network (FVN) (Wu et al. [2017]), maps nonlinear representations of images into a linearly-separable space, by reinforcing a LDA on top of the deep neural network. This Optimizes a LDA-based objective function, with stochastic gradient descent, and back-propagates LDA gradients, involved in Fisher vector encoding. It is due to the fact that, gradient of cross-entropy loss may help enlarging the inter-class differences, while it is unable to reduce the intra-class variations.

The Fisher AEN also does not use any Fisher kernels (like DFK), or is not trained by backpropagation on Fisher vectors (like HFV). Instead of involving Fisher vector encoding in the gradients (like FVN), the Fisher AEN backpropagates through standard formulation of gradient, generally employed by a classic AEN, to update the learning parameters.

Chapter 3

Fisher Discrimination

There is a vast amount of evidence, demonstrating the effectiveness of discriminant analysis methods for maximizing class separability. The Fisher's criterion has been extensively used in LDA and a wide variety of its extensions. For instance, Non-parametric LDA considers boundary information in inter-class scattering. The kernel version of LDA (Baudat and Anouar [2000]), handles a extreme nonlinearity of the sample set. There are also a number of methods, which aim to overcome singularity of the inverse intra-class covariance matrix of the mapped space. In a related development, dual subspaces are employed to construct LDA classifiers (Wang and Tang [2004]). This chapter starts with formulation of the conventional LDA for subspace mapping into low-dimensional spaces (dimension reduction), then proceeds to revisit this formulation for projection into high-dimensional spaces (dimension expansion), and finally ends up with the introduction of distributed Fisher discrimination.

3.1 Fisher Discrimination for Dimension Reduction

Suppose an input set $\mathcal{X} = \{(x_1, c_1), \ldots, (x_N, c_N)\}$, such that $(x_i, c_i) \in \mathbb{R}^d \times \mathcal{C}$ for all the $i \in [1, N]$ and $\mathcal{C}$ is a discrete set of classes. For a dimension reduction regime, the aim is to find a matrix $\mathbf{B} \in \mathbb{R}^{d \times c}$ that maps the input vector x_i onto the vector $y_i = \mathbf{B}^T x_i$ in a low-dimensional space $\mathcal{Y} = \{y_1, \ldots, y_N\}$, such that $y_i \in \mathbb{R}^c$ for all $i \in [1, N]$, conditioned on $c \ll d$. This mapping tries to maximize the separability between classes of the set $\mathcal{X}$ and minimize the scattering within them, using Fisher's criterion. One approach to compute this mapping, is supervised learning by LDA (Bishop [2006]).

The mapping matrix can be determined by maximizing the Fisher's criterion $\mathcal{J}_B$ given by,

$$\mathcal{J}_B = tr\left[\left(\mathbf{B}^T\mathbf{S}_{\mathbf{w}B}\mathbf{B}\right)^{-1}\left(\mathbf{B}^T\mathbf{S}_{\mathbf{b}B}\mathbf{B}\right)\right] \tag{3.1}$$

where $tr(.)$ is the diagonal summation operator. Here, the within-between class scatterings of set $\mathcal{S}_B = \{\mathbf{S}_{\mathbf{w}B}, \mathbf{S}_{\mathbf{b}B}\}$ are defined as,

$$\mathbf{S}_{\mathbf{w}B} = \sum_{j=1}^{c} \sum_{x_i \in \mathbf{C}_j} (x_i - \mu_j)(x_i - \mu_j)^T \tag{3.2}$$

$$\mathbf{S}_{\mathbf{b}B} = \sum_{j=1}^{c} (\mu_j - \bar{\mu})(\mu_j - \bar{\mu})^T \tag{3.3}$$

where c, μ_j and $\bar{\mu}$ are number of classes, mean over class $\mathbf{C}_j$, and mean over the set $\mathcal{X}$. The $\mathbf{S}_{\mathbf{w}B} \in \mathbb{R}^{d\times d}$ can be regarded as the average class-specific covariance, whereas the $\mathbf{S}_{\mathbf{b}B} \in \mathbb{R}^{d\times d}$ can be viewed as the mean distance between all different classes. The purpose of $\mathcal{J}_B$ is to maximize the between-class scatter, while preserving within-class dispersion.

The solution can be retrieved from a generalized eigenvalue problem $\mathbf{S}_{\mathbf{b}B}\mathbf{B} = \lambda \mathbf{S}_{\mathbf{w}B}\mathbf{B}$. Since the rank of $\mathbf{S}_{\mathbf{b}B}$ is $c-1$, the solution for c classes, is eigenvectors corresponding to the largest c eigenvalues of $\mathbf{S}_{\mathbf{w}B}^{-1}\mathbf{S}_{\mathbf{b}B}$, subject to $c \ll d$ (Fukunaga [1990]). To prove this solution, consider the matrix $\mathbf{B} \in \mathbb{R}^{d\times c}$ that maps a d-dimensional space $\mathcal{X}$ to a c-dimensional space $\mathcal{Y}$,

$$\mathcal{Y} = \mathbf{B}^T\mathcal{X} \tag{3.4}$$

subject to the fact that, $\mathcal{X}$ and $\mathcal{Y}$ are linearly independent, but not necessarily orthogonal. Since the scatter matrices in the $\mathcal{X}$-space i.e. $\mathbf{S}_{\mathbf{w}\mathcal{X}} = \mathbf{S}_{\mathbf{w}B}$ and $\mathbf{S}_{\mathbf{b}\mathcal{X}} = \mathbf{S}_{\mathbf{b}B}$, are of the form of a covariance matrix, the scatterings of $\mathcal{Y}$-space can be expressed as follows,

$$\begin{aligned}\mathbf{S}_{\mathbf{w}\mathcal{Y}} &= \mathbf{B}^T\mathbf{S}_{\mathbf{w}\mathcal{X}}\mathbf{B}\\ \mathbf{S}_{\mathbf{b}\mathcal{Y}} &= \mathbf{B}^T\mathbf{S}_{\mathbf{b}\mathcal{X}}\mathbf{B}\end{aligned} \tag{3.5}$$

Now, Equation 3.1 can be reformulated as,

$$\mathcal{J}_B = tr(\mathbf{S}_{\mathbf{w}\mathcal{Y}}^{-1}\mathbf{S}_{\mathbf{b}\mathcal{Y}}) = tr\Big[\big(\mathbf{B}^T\mathbf{S}_{\mathbf{w}\mathcal{X}}\mathbf{B}\big)^{-1}\big(\mathbf{B}^T\mathbf{S}_{\mathbf{b}\mathcal{X}}\mathbf{B}\big)\Big] \tag{3.6}$$

To maximize Equation 3.6, its derivative with respect to **B** can be set to zero.

$$\frac{\delta\mathcal{J}_B}{\delta\mathbf{B}} = -2\mathbf{S}_{\mathbf{w}\mathcal{X}}\mathbf{B}\mathbf{S}_{\mathbf{w}\mathcal{Y}}^{-1}\mathbf{S}_{\mathbf{b}\mathcal{Y}}\mathbf{S}_{\mathbf{w}\mathcal{Y}}^{-1} + 2\mathbf{S}_{\mathbf{b}\mathcal{X}}\mathbf{B}\mathbf{S}_{\mathbf{w}\mathcal{Y}}^{-1} = 0 \tag{3.7}$$

Then, the optimum **B** should satisfy,

$$(\mathbf{S}_{\mathbf{w}\mathcal{X}}^{-1}\mathbf{S}_{\mathbf{b}\mathcal{X}})\mathbf{B} = \mathbf{B}(\mathbf{S}_{\mathbf{w}\mathcal{Y}}^{-1}\mathbf{S}_{\mathbf{b}\mathcal{Y}}) \tag{3.8}$$

It is possible to diagonalize $\mathbf{S}_{\mathbf{b}\mathcal{Y}}$ and $\mathbf{S}_{\mathbf{w}\mathcal{Y}}$ to $\mathbf{\Lambda}_c$ and $\mathbf{\Omega}_c$, using a linear transformation $\mathbf{Z} = \mathbf{C}\mathcal{X}$, such that $\mathbf{C} \in \mathbb{R}^{c\times c}$ is a non-singular square matrix and its inverse exists.

$$\begin{aligned}\mathbf{\Lambda}_c &= \mathbf{C}^T\mathbf{S}_{\mathbf{b}\mathcal{Y}}\mathbf{C}\\ \mathbf{\Omega}_c &= \mathbf{C}^T\mathbf{S}_{\mathbf{w}\mathcal{Y}}\mathbf{C}\end{aligned} \tag{3.9}$$

The Fisher's criterion $\mathcal{J}_B$, remains invariant under this linear transformation because,

$$\begin{aligned}tr(\mathbf{S}_{\mathbf{w}Z}^{-1}\mathbf{S}_{\mathbf{b}Z}) &= tr\big[\big(\mathbf{C}^T\mathbf{S}_{\mathbf{w}\mathcal{Y}}\mathbf{C}\big)^{-1}\big(\mathbf{C}^T\mathbf{S}_{\mathbf{b}\mathcal{Y}}\mathbf{B}\big)\big]\\ &= tr\big(\mathbf{C}^{-1}\mathbf{S}_{\mathbf{w}\mathcal{Y}}^{-1}\big(\mathbf{C}^T\big)^{-1}\mathbf{C}^T\mathbf{S}_{\mathbf{b}\mathcal{Y}}\mathbf{C}\big)\\ &= tr\big(\mathbf{S}_{\mathbf{w}\mathcal{Y}}^{-1}\mathbf{S}_{\mathbf{b}\mathcal{Y}}\mathbf{C}\mathbf{C}^{-1}\big)\\ &= tr\big(\mathbf{S}_{\mathbf{w}\mathcal{Y}}^{-1}\mathbf{S}_{\mathbf{b}\mathcal{Y}}\big)\end{aligned} \tag{3.10}$$

Considering Equation 3.9, the Equation 3.8 can be rewritten as follows,

$$(\mathbf{S}_{\mathbf{w}\mathcal{X}}^{-1}\mathbf{S}_{\mathbf{b}\mathcal{X}})\mathbf{B} = \mathbf{B}(\mathbf{C}\mathbf{\Lambda}_c\mathbf{C}^{-1}) \tag{3.11}$$

Reformulation of Equation 3.11 by using a minor matrix operation gives,

$$(\mathbf{S}_{\mathbf{w}\mathcal{X}}^{-1}\mathbf{S}_{\mathbf{b}\mathcal{X}})\mathbf{BC} = \mathbf{BC}\mathbf{\Lambda}_c \tag{3.12}$$

which means that $\mathbf{\Lambda}_c$ contains d eigenvalues of $\mathbf{S}_{\mathbf{w}\mathcal{X}}^{-1}\mathbf{S}_{\mathbf{b}\mathcal{X}}$ and $\mathbf{BC}$ corresponds to c eigenvectors.

The trace of a matrix equals to the summation of its eigenvalues,

$$\mathcal{J}_B = tr(\mathbf{S}_{\mathbf{w}\mathcal{Y}}^{-1}\mathbf{S}_{\mathbf{b}\mathcal{Y}}) = \lambda_1 + \cdots + \lambda_c \tag{3.13}$$

and hence, $\mathcal{J}_B$ can be maximized by selecting the largest c eigenvalues corresponding to c eigenvectors of $\mathbf{S}_{\mathbf{w}B}^{-1}\mathbf{S}_{\mathbf{b}B}$. This proves that, mapping of the set $\mathcal{X}$ into the c eigenvectors of $\mathbf{S}_{\mathbf{w}B}^{-1}\mathbf{S}_{\mathbf{b}B}$, forms a c-dimensional subspace ($\mathcal{Y}$) spanned by these c eigenvectors. The Fisher's criterion $\mathcal{J}_B$ also equals the summation of c largest eigenvalues. According to Equation 3.10, this criterion is invariable to the selection of the coordinate system in the mapped subspace.

3.2 Fisher Discrimination for Dimension Expansion

Suppose that the set $\mathcal{S}_B = \{\mathbf{S}_{\mathbf{w}B}, \mathbf{S}_{\mathbf{b}B}\}$ contains non-zero scattering matrices, that both of them cannot be identity matrix, at the same time. Otherwise, the permutation of trace operator in Equation 3.1, implies infinite valid orthogonal solutions. By applying cyclic permutation and imposing orthogonality through $\mathbf{B}^T\mathbf{B} = \mathbf{I}$, the $\mathcal{J}_B$ becomes,

$$\begin{aligned}
\mathcal{J}_B &= tr\big[\big(\mathbf{B}^T\mathbf{S}_{\mathbf{w}B}\mathbf{B}\big)^{-1}\big(\mathbf{B}^T\mathbf{S}_{\mathbf{b}B}\mathbf{B}\big)\big] \\
&= tr\big(\mathbf{B}^{-1}\mathbf{S}_{\mathbf{w}B}^{-1}\big(\mathbf{B}^T\big)^{-1}\mathbf{B}^T\mathbf{S}_{\mathbf{b}B}\mathbf{B}\big) \\
&= tr\big(\mathbf{S}_{\mathbf{w}B}^{-1}\mathbf{S}_{\mathbf{b}B}\mathbf{B}\mathbf{B}^{-1}\big) \\
&= tr\big(\mathbf{S}_{\mathbf{w}B}^{-1}\,\mathbf{S}_{\mathbf{b}B}\big)
\end{aligned} \tag{3.14}$$

For a dimension expansion regime, consider a projection matrix $\mathbf{A} \in \mathbb{R}^{d\times c}$ conditioned at $c > d$. Then, it is required to define a new inter-class scattering $\mathbf{S}_{\mathbf{w}A} \in \mathbb{R}^{c\times c}$, such that,

$$tr(\mathbf{S}_{\mathbf{w}B}) = tr(\mathbf{S}_{\mathbf{w}A}) \tag{3.15}$$

Note that in Equations 3.2, the summation is defined over all classes $(j \in [1, c])$ of the set $\mathcal{X}$. To satisfy Equation 3.15, $\mathbf{S}_{\mathbf{w}A}$ can be considered as a square matrix, of size $c \times c$, such that, all of its entries except the main diagonal ones, are set to zero. Hence, the jth diagonal entry is defined as follows,

$$\mathbf{S}_{\mathbf{w}A}(j,j) = tr\left[\sum_{x_i \in \mathbf{C}_j} (x_i - \mu_j)(x_i - \mu_j)^T\right] \quad \forall\, j \in [1, c] \tag{3.16}$$

From Equations 3.15 and the similarity invariance of the trace operator, $\mathbf{S}_{\mathbf{w}B}$ and $\mathbf{S}_{\mathbf{w}A}$ are similar matrices (Horn and Johnson [2012]). This implies that, there should exist a non-singular matrix $\mathbf{\Gamma}_w$, such that,

$$\mathbf{S}_{\mathbf{w}B} = \mathbf{\Gamma}_w^{-1} \mathbf{S}_{\mathbf{w}A} \mathbf{\Gamma}_w \tag{3.17}$$

By minor matrix operations, Equation 3.17 can be formulated as,

$$\mathbf{\Gamma}_w \mathbf{S}_{\mathbf{w}B} - \mathbf{S}_{\mathbf{w}A} \mathbf{\Gamma}_w = \mathbf{0} \tag{3.18}$$

which is a special case of the Sylvester equation (Lee and Vu [2011]) and can be solved for $\mathbf{\Gamma}_w$ by either Kronecker tensor trick or using generalized eigen-decomposition. This is because $\mathbf{S}_{\mathbf{w}B}$ and $\mathbf{S}_{\mathbf{w}A}$ are non-singular matrices. The closed form solution for Equation 3.18 is,

$$vec(\mathbf{\Gamma}_w) = \mathbf{I} \otimes (-\mathbf{S}_{\mathbf{w}A}) - \mathbf{S}_{\mathbf{w}B}^{\mathbf{T}} \otimes \mathbf{I} \tag{3.19}$$

where $vec(.)$ is the vectorization operator and $\otimes$ is Kronecker product. With the same reasoning, $\mathbf{S}_{\mathbf{b}A}$ can be defined as a square matrix, of size $c \times c$, such that,

$$tr(\mathbf{S}_{\mathbf{b}B}) = tr(\mathbf{S}_{\mathbf{b}A}) \tag{3.20}$$

and there should exist a non-singular matrix $\mathbf{\Gamma}_b$ which gives,

$$\mathbf{S}_{\mathbf{b}B} = \mathbf{\Gamma}_b^{-1} \mathbf{S}_{\mathbf{b}A} \mathbf{\Gamma}_b \tag{3.21}$$

On the other hand, Equations 3.17 and 3.21 hold,

$$\begin{aligned}\mathbf{S}_{\mathbf{w}B}^{-1}\mathbf{S}_{\mathbf{b}B} &= \left(\boldsymbol{\Gamma}_w^{-1}\mathbf{S}_{\mathbf{w}A}\boldsymbol{\Gamma}_w\right)^{-1}\left(\boldsymbol{\Gamma}_b^{-1}\mathbf{S}_{\mathbf{b}A}\boldsymbol{\Gamma}_b\right) \\ &= \boldsymbol{\Gamma}_w^{-1}\mathbf{S}_{\mathbf{w}A}^{-1}\boldsymbol{\Gamma}_w\boldsymbol{\Gamma}_b^{-1}\mathbf{S}_{\mathbf{b}A}\boldsymbol{\Gamma}_b \qquad (3.22)\end{aligned}$$

Due to the similarity invariance in Equations 3.15 and 3.20, the cyclic permutation of the trace operator can be considered to set,

$$\boldsymbol{\Gamma}_b = \boldsymbol{\Gamma}_w \qquad (3.23)$$

As a result, Equation 3.21 implies $\mathbf{S}_{\mathbf{b}A}$ as,

$$\mathbf{S}_{\mathbf{b}A} = \boldsymbol{\Gamma}_b\mathbf{S}_{\mathbf{b}B}\boldsymbol{\Gamma}_b^{-1} \qquad (3.24)$$

Now, Equation 3.22 can be worked out by substitution of $\boldsymbol{\Gamma}_b$ with $\boldsymbol{\Gamma}_w$ (Equation 3.23)

$$\begin{aligned}\mathbf{S}_{\mathbf{w}B}^{-1}\mathbf{S}_{\mathbf{b}B} &= \boldsymbol{\Gamma}_w^{-1}\mathbf{S}_{\mathbf{w}A}^{-1}(\mathbf{I})\mathbf{S}_{\mathbf{b}A}\boldsymbol{\Gamma}_b \\ &= \boldsymbol{\Gamma}_w^{-1}\left(\mathbf{S}_{\mathbf{w}A}^{-1}\mathbf{S}_{\mathbf{b}A}\right)\boldsymbol{\Gamma}_w \qquad (3.25)\end{aligned}$$

This proves that $\mathbf{S}_{\mathbf{w}B}^{-1}\mathbf{S}_{\mathbf{b}B}$ and $\mathbf{S}_{\mathbf{w}A}^{-1}\mathbf{S}_{\mathbf{b}A}$ are similar matrices, that gives,

$$tr\left(\mathbf{S}_{\mathbf{w}B}^{-1}\mathbf{S}_{\mathbf{b}B}\right) = tr\left(\mathbf{S}_{\mathbf{w}A}^{-1}\mathbf{S}_{\mathbf{b}A}\right) \qquad (3.26)$$

Looking back at Equation 3.14, a new optimization problem can be formulated, based on $\mathbf{S}_{\mathbf{w}A}$ and $\mathbf{S}_{\mathbf{b}A}$ as follows,

$$\mathcal{J}_A = tr\left[\left(\mathbf{A}\mathbf{S}_{\mathbf{w}A}\mathbf{A}^T\right)^{-1}\left(\mathbf{A}\mathbf{S}_{\mathbf{b}A}\mathbf{A}^T\right)\right] \qquad (3.27)$$

Equation 3.27 inherits the discrimination power and orthogonality of Equation 3.1, but in contrast, this is aligned with the number of classes (c) instead of the dimension of input (d).

3.3 Distributed Fisher Discrimination

Suppose that a set of N objective functions $\mathcal{J} = \{\mathcal{J}_1, \ldots, \mathcal{J}_N\}$ are optimized to map an input set $\mathcal{X} = \{X_1, \ldots, X_N\}$ through a projection set $\mathcal{A} = \{\mathbf{A}_1, \ldots, \mathbf{A}_N\}$ into an output set $\mathcal{Y} = \{Y_1, \ldots, Y_N\}$. This holds $Y_k = X_k\mathbf{A}_k$ and each Fisher's criterion $\mathcal{J}_k$ will get maximized. The aim is to prove that concatenation of all entries in the set $\mathcal{Y}$ is also well-discriminated, based on Fisher's criterion. This implies that Fisher discrimination can be successfully applied for N partitions of data or N sets of features, in any distributed learning paradigms.

The scattering matrices of set $\mathcal{Y}$ can be calculated, using distributed projections $\{Y_1, \ldots, Y_N\}$ as follows (Valcarcel Macua et al. [2011]),

$$\begin{aligned} \mathbf{S_w} &= \frac{1}{N}\sum_{k=1}^{N} \bar{Y}_k \bar{Y}_k^T \\ \mathbf{S_b} &= \sum_{j=1}^{c} (\mathbf{M}_j - \bar{\mathbf{M}})(\mathbf{M}_j - \bar{\mathbf{M}})^T \end{aligned} \tag{3.28}$$

such that $\mathbf{M}_j = \frac{1}{N}\sum_{k=1}^{N} \mathbf{M}_{jk}$ and $\bar{\mathbf{M}} = \frac{1}{N}\sum_{k=1}^{N} \bar{\mathbf{M}}_k$. They are mean over the jth class and mean over all representations of the set $\mathcal{Y}$. It is known that the within-between class scatterings for each $Y_k \in \mathcal{Y}$ are defined as,

$$\begin{aligned} \mathbf{S}_{\mathbf{w}k} &= \sum_{j=1}^{c} \sum_{y_{ik} \in \mathbf{C}_j} (y_{ik} - \mu_{jk})(y_{ik} - \mu_{jk})^T \\ \mathbf{S}_{\mathbf{b}k} &= \sum_{j=1}^{c} (\mu_{jk} - \bar{\mu}_k)(\mu_{jk} - \bar{\mu}_k)^T \end{aligned} \tag{3.29}$$

where y_{ik}, c, μ_{jk} and $\bar{\mu}_k$ are; vectors in Y_k, number of classes, mean over class $\mathbf{C}_j$ and mean over all samples in Y_k. These scatterings also minimize $\mathcal{J}_k$ as,

$$\mathcal{J}_k = -log\bigg(tr(\mathbf{S}_{\mathbf{w}k}) \big[tr(\mathbf{S}_{\mathbf{b}k}) \big]^{-1} \bigg) \tag{3.30}$$

Interchanging the order of the summations in Equation 3.28 and considering Equation 3.29, the following holds,

$$\begin{aligned} \mathbf{S_w} &= \frac{1}{N}\sum_{k=1}^{N}\mathbf{S}_{\mathbf{w}k} \\ \mathbf{S_b} &= \frac{1}{N}\sum_{k=1}^{N}\mathbf{S}_{\mathbf{b}k} \end{aligned} \tag{3.31}$$

Now, the $tr(.)$ operator can be applied to the both sides of Equation 3.31 to give,

$$\begin{aligned} tr(\mathbf{S_w}) &= \sum_{k=1}^{N} tr\left(\frac{\mathbf{S}_{\mathbf{w}k}}{N}\right) \\ tr(\mathbf{S_b}) &= \sum_{k=1}^{N} tr\left(\frac{\mathbf{S}_{\mathbf{b}k}}{N}\right) \end{aligned} \tag{3.32}$$

The Arithmetic Mean-Geometric Mean inequality ([Steele, 2004]) implies that,

$$\begin{aligned} \sum_{k=1}^{N} tr\left(\frac{\mathbf{S}_{\mathbf{w}k}}{N}\right) &\geqslant \left[\prod_{k=1}^{N} tr(\mathbf{S}_{\mathbf{w}k})\right]^{\frac{1}{N}} \\ \sum_{k=1}^{N} tr\left(\frac{\mathbf{S}_{\mathbf{b}k}}{N}\right) &\geqslant \left[\prod_{k=1}^{N} tr(\mathbf{S}_{\mathbf{b}k})\right]^{\frac{1}{N}} \end{aligned} \tag{3.33}$$

which results,

$$\frac{tr(\mathbf{S_w})}{tr(\mathbf{S_b})} \geqslant \left[\prod_{k=1}^{N}\frac{tr(\mathbf{S}_{\mathbf{w}k})}{tr(\mathbf{S}_{\mathbf{b}k})}\right]^{\frac{1}{N}} \tag{3.34}$$

Applying negative logarithm to both sides of Equation 3.34 produces,

$$-log\left(tr(\mathbf{S_w})\left[tr(\mathbf{S_b})\right]^{-1}\right) \leqslant -\frac{1}{N}\sum_{k=1}^{N} log\left(tr(\mathbf{S}_{\mathbf{w}k})\left[tr(\mathbf{S}_{\mathbf{b}k})\right]^{-1}\right) \tag{3.35}$$

Considering Equation 3.30, this indicates that,

$$\mathcal{J} \leqslant \frac{1}{N}\sum_{k=1}^{N}\mathcal{J}_k \qquad (3.36)$$

This means that the minimum separability of the set $\mathcal{Y}$, has an upper bound which is equal to the average of all minimum discriminations, imposed independently into $\{Y_1, \dots, Y_N\}$ by $\{\mathbf{A}_1, \dots, \mathbf{A}_N\}$. In other words, solving N distributed local Fisher discrimination problems $\{\mathcal{J}_1, \dots, \mathcal{J}_N\}$, can generate a general solution for the global discrimination problem $\mathcal{J}$ over the set $\mathcal{X}$, when all N projections are concatenated. This finding leads to introduce some novel distributed deep architectures in this thesis, trained by Deep Fisher Discriminant Learning.

3.4 Conclusion

In this chapter, the idea of Fisher discrimination was mathematically formulated. All the entities and parameters of interest towards achieving optimal classification by maximizing Fisher's criterion, were also clarified. The key was to determine an associated transform matrix to better distinguish the classes by maximizing the scattering between relative to that within them. A novel dimension expansion method as a dual to conventional dimension reduction, was proposed to further enhance the discriminative power of the classification by learning a projection matrix to map into higher dimensions same as initial number of classes, inspired by Fisher's criterion. Built upon these developments, a distributed Fisher discrimination technique was introduced to cope with a large number of classes. It was shown that the overall Fisher discrimination problem was bounded by the sum of Fisher discriminations of each classes. In other words, the global problem could be solved when each of the local discrimination problems was solved independently.

Part I

Semantic Segmentation

Chapter 4

Supervised Colour Transformation

Colour is a fundamental descriptor for numerous tasks in computer vision (Van De Sande et al. [2010]). For these applications, colour space and descriptor are important (Chong et al. [2008]) and hence, selection of the proper colour transformation, is critical to achieve high performances (Strutz [2012]). For example, texture and complex shapes are better handled by RGB colour space (Hu et al. [2011b]), while CIE, LUV or Lab spaces (Wyszecki and Stiles [1982]) are often used for segmentation (Meyer and Greenberg [1980]).

The challenge of semantic segmentation has generally been addressed by solving a pixelwise classification problem. The aim is to predict the corresponding class of each pixel in a scene, by assigning a predefined label. The approach is to propose a supervised colour transformation. Almost all colour transformations, in the literature and standards, are derived by perceptual criteria rather than image information. Therefore, the proposed supervised colour transformation is a novel approach to find a new colour space, based on class information, for the application at hand. This transformation is computed via supervised learning, and is employed as a general preprocessing step. For the purpose of semantic segmentation, it can be implemented before feature extraction, in a standard pipeline of pixelwise classification.

For implementation, class information in the dataset is employed to project the initial colour space to a new space, spanned by the number of classes (dimension expansion). This allows for the maximum pairwise distances between classes and the minimum scattering within each class. A subspace projection is deployed to return back to a new colour space with the same dimension as the primary colour space (dimension reduction). It also preserves the class separability, imposed by the first projection.

This projection-backprojection is the backbone of the supervised colour transformation. It is learned from the distribution of data and classes in the dataset (projection). It is also exposed the acquired distinction in the form of another colour space (backprojection). For the sake of projection, FDA is reformulated (Section 3.2), to address the mapping from a low-dimensional space (colour) to a high-dimensional space (class). For the backprojection, metric MDS is used, which preserves geodesic distances between the classes in the high-dimensional space (class), while mapping back to another low-dimensional space (colour).

4.1 Method

To achieve better performance on pixelwise classification, the Fisher discrimination is utilized to map a primary colour space into a target colour space. The latter has the same dimension as the former, but represents higher class separability. This transformation consists of a projection from primary colour space to a space, spanned by the number of classes (dimension expansion), followed by a backprojection from this space to the target colour space (dimension reduction). Finally, an anisotropic scalings-translations is applied to fit the target colour cube in the acceptable standard range of the primary colour space.

The lack of between-class scattering specificity (Loog et al. [2001]), is compounded by the burden of dimensionality. To deal with the high-dimensional data, a number of approaches have been proposed. These are independence rule (Bickel and Levina [2004]), feature annealed independence rule (Fan and Fan [2008]), and NSC classifier (Tibshirani et al. [2002]).

Revisiting classical FDA, the proposed projection aims to map the primary colour channels to a higher-dimensional class space. This is in contrast with approaches, such as, LDA (McLachlan [2004]), where class-specific covariance is used to define within-class scattering, while between-class scatter is considered to be uniform for distinct classes. Then, the backprojection tries to map the high-dimensional class space into the low-dimensional target space, using MDS to preserve the distances imposed by the above projection. The result is a highly distinct colour representation, that benefits the feature extraction process in classification pipeline.

For evaluation purpose, several experiments are conducted on two semantic segmentation algorithms and two datasets, which are both publicly available. First, TextonBoost (Krähenbühl and Koltun [2012]) is employed on MSRC-21 dataset (Shotton et al. [2009]). This uses colour, histogram of oriented gradients (HOG), and pixel location features. Second, Darwin (Gould [2012]) is deployed on SBD dataset (Gould et al. [2009]), that employs RGB colour of the pixel, dense HOG, LBP-like features, and averages over image rows and columns. The supervised colour transformation is applied to preprocess images, feeding a semantic segmentation process. The outcomes show that the proposed approach consistently improves the overall and average precisions of both, TextonBoost and Darwin algorithms for pixelwise semantic segmentation.

4.2 Formulation

In order to project the primary colour space to the class space (dimension expansion), the Fisher's criterion is minimized and an orthogonality constraint is imposed to compute the projection matrix. Then, a subspace mapping matrix is computed, which backprojects the class space into the target space (dimension reduction).

4.2.1 Projection for Dimension Expansion

Suppose that an input $\mathcal{X} \in \mathbb{R}^{n \times d}$ consists of n pixels, each with d colour components. This is presented to a projection matrix $\mathbf{A} \in \mathbb{R}^{d \times c}$ which maps it into $\mathcal{P} = \mathcal{X}\mathbf{A}$ where $\mathcal{P} \in \mathbb{R}^{n \times c}$. Here, c is the number of semantic classes in the $\mathcal{X}$, such that, $c > d$. To figure out the projection matrix $\mathbf{A}$, the Fisher's criterion (Bishop [2006]), *i.e.* the ratio of inter-intra class scatterings of set $\mathcal{S}_A = \{\mathbf{S}_{\mathbf{w}A}, \mathbf{S}_{\mathbf{b}A}\}$ is minimized by solving,

$$\arg\min_{\mathbf{A}\mathbf{A}^T=\mathbf{I}} \mathcal{H}(\mathbf{A}) = tr\big(\mathbf{A}\mathbf{S}_{\mathbf{w}A}\mathbf{A}^T\big)\left[tr\big(\mathbf{A}\mathbf{S}_{\mathbf{b}A}\mathbf{A}^T\big)\right]^{-1} \tag{4.1}$$

Here, $tr(.)$ is the trace operator and $\mathbf{I}$ indicates the identity matrix. This equation makes the highest possible separability among classes, whilst the constraint term, $\mathbf{A}\mathbf{A}^T = \mathbf{I}$, imposes orthogonality into the projection matrix $\mathbf{A}$.

For consistency of dimensionality in matrix operations of Equation 4.1, the scattering set $\mathcal{S}_A$ should belong to $\mathbb{R}^{c\times c}$ but, classical definitions of scattering implies $\mathcal{S}_A \in \mathbb{R}^{d\times d}$ because, it is currently aligned with the depth of $\mathcal{X}$. It is not possible to employ the linear discriminant analysis (Fukunaga [1990]) to solve Equation 4.1, because this is no longer a dimension reduction problem. In contrast, this optimization problem tries to increase the dimension of input (d) by projecting to a higher dimension (c), where $c > d$.

The solution for this inconsistency is redefinition of the scattering set $\mathcal{S}_A$ based on the number of classes (c) rather than the dimension of input (d). Starting from a scattering set $\mathcal{S}_B = \{\mathbf{S_w}_B, \mathbf{S_b}_B\}$ which $\mathcal{S}_B \in \mathbb{R}^{d\times d}$, within and between-class scatterings are,

$$\mathbf{S_w}_B = \sum_{j=1}^{c} \sum_{x_i \in \mathbf{C}_j} (x_i - \mu_j)(x_i - \mu_j)^T \tag{4.2}$$

$$\mathbf{S_b}_B = \sum_{j=1}^{c} (\mu_j - \bar{\mu})(\mu_j - \bar{\mu})^T \tag{4.3}$$

where x_i, c, μ_j and $\bar{\mu}$ are the input samples, number of classes, mean over class $\mathbf{C}_j$ and mean over $\mathcal{X}$, respectively. Now, the new scattering set $\mathcal{S}_A \in \mathbb{R}^{c\times c}$ can be defined. Assume $\mathbf{S_w}_A$ as a square matrix, of size $c \times c$, such that all of its entries, except main diagonal ones, are set to zero. If the jth entry is formulated as,

$$\mathbf{S_w}_A(j,j) = tr\left[\sum_{x_i \in \mathbf{C}_j} (x_i - \mu_j)(x_i - \mu_j)^T \right] \quad \forall\, j \in [1, c] \tag{4.4}$$

then, a non-singular matrix $\mathbf{\Gamma}_w$ is available to satisfy the following equation,

$$vec(\mathbf{\Gamma}_w) = \mathbf{I} \otimes (-\mathbf{S_w}_A) - \mathbf{S}^{\mathbf{T}}_{\mathbf{w}B} \otimes \mathbf{I} \tag{4.5}$$

which $vec(.)$ is a vectorization operator and $\otimes$ is Kronecker product.

Equation 4.5 is a closed-form formulation of Sylvester equation (Lee and Vu [2011]) for $\mathbf{\Gamma}_w$, that can be solved by either Kronecker tensor trick or generalized eigen decomposition. Setting $\mathbf{\Gamma}_b = \mathbf{\Gamma}_w$ implies that,

$$\mathbf{S_b}_A = \mathbf{\Gamma}_b \mathbf{S_b}_B \mathbf{\Gamma}_b^{-1} \tag{4.6}$$

Algorithm 1 Supervised Projection

```
Input: input $\mathcal{X} \in \mathbb{R}^{n \times d}$ with $n$ pixels of depth $d$
Output: optimal projection matrix $\mathbf{A} \in \mathbb{R}^{d \times c}$

1: Compute $\mathbf{S_w}_A$ (Equation 4.4) and $\mathbf{S_b}_A$ (Equation 4.6)
2: Set initialization point $\mathbf{A}^{(0)}$ as $c$ largest eigenvalues of $\mathbf{S}^{-1}_{\mathbf{w}A}\mathbf{S_b}_A$
3: Solve Equation 4.1 by nonlinear least-squares minimization algorithm
```

It is shown that, the solution can be retrieved by set of eigenvectors corresponding to the largest c eigenvalues of $\mathbf{S}^{-1}_{\mathbf{w}A}\mathbf{S_b}_A$ (Section 3.2). Although this solution can be considered as a suboptimal projection matrix, it is only employed as an initialization point $\mathbf{A}^{(0)}$ to solve Equation 4.1 by a nonlinear least-squares optimization method. In general, Fisher's criterion is in form of trace-of-quotient, which can be solved by a generalized eigenvalue method (Bishop [2006]), but Equation 4.1 is arranged as quotient-of-trace, that requires a different solution (Cunningham and Ghahramani [2015]).

With the aid of the information described above, and the Matlab optimization toolbox (Coleman and Li [1996]), least-squares minimization with trust-region-reflective is used to solve Equation 4.1. Algorithm 1 summarizes the computing of projection matrix **A**.

4.2.2 Backprojection for Dimension Reduction

The backprojection preserves pairwise cluster distances of the class space ($\mathcal{P} = \mathcal{X}\mathbf{A}$), derived from the learned projection matrix **A**. This naturally leads to application of linear and nonlinear embedding techniques of dimensionality reduction, that attempt to preserve global or local properties of the original data, in low-dimensional representations. Metric MDS is the subspace mapping algorithm of choice for this end, because it has the capacity to preserve pairwise distances in the class space, and to employ a wide variety of loss functions. In this situation, the stress function is employed to measure the error between the pairwise distances.

To go further, the backprojection cost function $\mathcal{Q}(\mathbf{B})$ becomes,

$$\mathcal{Q}(\mathbf{B}) = -\sum_{x_i \in \mathcal{X}} \left(||\mathcal{P}||^2 - ||\mathcal{P}\mathbf{B}||^2 \right)^2 \tag{4.7}$$

where $||\cdot||$ is the vector norm and $\mathcal{Y} = \mathcal{P}\mathbf{B}$ is the target space. Here, the terms $||\mathcal{P}||$ and $||\mathcal{P}\mathbf{B}||$ are the Euclidean distances in the class and target spaces.

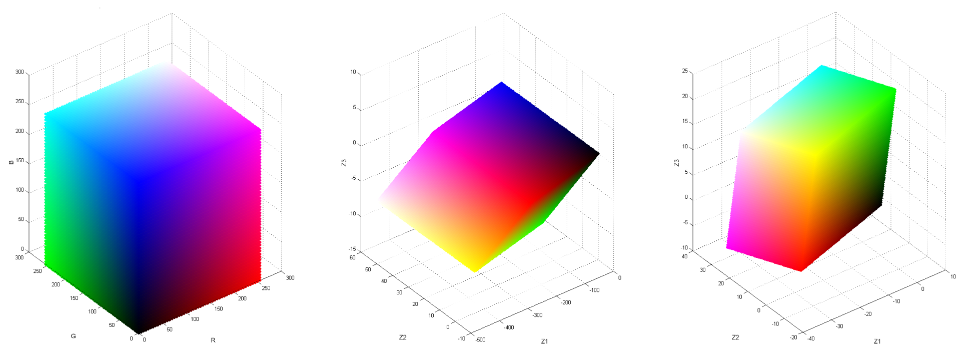

Figure 4.1: Standard RGB and transformed colour cubes for MSRC-21 and SBD datasets. The standard colour spectrum is mapped differently, according to the specification of the datasets under study. Since the transformed cubes do not comply with the typical ranges of a standard colour space, they should be scaled and translated by an anisotropic transformation.

As a result, the separation between pixel pairs belonging to different classes, is maximized by the projection matrix **A** and hence, the backprojection matrix **B** is expected to preserve these distances. Moreover, the minimization of $\mathcal{Q}(\mathbf{B})$ can be performed, using various methods, such as, eigen-decomposition, or pseudo-Newton minimization, or conjugate gradient. In this instance, the conjugate gradient method is employed.

4.2.3 Anisotropic Scaling-Translation

The proposed supervised colour transformation, probably generates out-of-range values for $\mathcal{Y}$ as the target colour space (for example negative for the primary RGB colour space). To accommodate the requirements of primary colour spaces (RGB, LUV, Lab, etc.), the transformed colour cubes are scale-translated by,

$$\mathcal{Y}^* = \mathcal{Y}\mathbf{S} + \mathbf{T} \tag{4.8}$$

where **S** and **T** are anisotropic scaling and translation matrices. This is a straightforward scaling-translation operation, in the transformed target space. Further, these matrices are computed by the deploying of vertices in the colour cubes.

This is done by solving a linear equation, where six vertices are used to obtain six degrees of freedom. The degrees of freedom comprise three diagonal elements of each of the above two matrices. Figure 4.1 shows the transformed colour cubes for MSRC-21 and SBD datasets. It can be seen that, they do not comply with the standards of RGB as the primary colour space, because they show completely different colour settings, such as, negative values, learned from the experimental datasets.

4.3 Experiments

To conduct experiments on semantic segmentation, a pixelwise classification pipeline is considered. This employs the proposed colour transformation as a preprocessing step, such that, a raw input image is fed in, and the colour-transformed output goes to a pixelwise classifier. The experimental setup covers various classifiers and number of classes.

For the first experiment, TextonBoost framework (Krähenbühl and Koltun [2012]) is used to classify MSRC-21 dataset (Shotton et al. [2009]). This experiment provides the unary potentials for each class as the output of classification. The second experiment employs Darwin implementation (Gould [2012]) for Stanford Background Dataset (SBD) (Gould et al. [2009]), and delivers both unary and pairwise potentials.

The whole training set is used to learn the proposed supervised colour transformation by Algorithm 1. To apply MDS for the backprojection, Markov Chain Monte Carlo (MCMC) method is employed to subsample a balanced distribution of data among classes and then, to minimize Equation 4.7. This is finally followed by the anisotropic scaling-translation to form the target colour space. For comparison with other standard colour spaces, the Lab colour space is taken as the baseline, since it is employed by both TextonBoost and Darwin frameworks.

The other colour spaces are YCrCb, HSL, LUV (Wyszecki and Stiles [1982]), I1I2I3 (Geusebroek et al. [2001]), and O1O2O3 (Van De Sande et al. [2010]). Since the above classifiers work on the luminance channel, I1 from I1I2I3 and O3 from O1O2O3 canonical colour spaces, are deployed for the sake of comparison. The following subspace mapping methods: Heteroscedastic Discriminant Analysis (HDA) (Loog et al. [2001]); Maximum Margin Criterion (MMC); Singular Value Decomposition (SVD); and Kernel Principal Component Analysis (KPCA), are used as alternatives to MDS.

4.3.1 Datasets

The MSRC-21 dataset (Shotton et al. [2009]), consists of 591 colour images, of size 320 $\times$ 213 pixels, with corresponding ground-truth segmentation for 21 object classes. The evaluation protocol considers 276 images for training and 256 images for test (Krähenbühl and Koltun [2012]). The SBD (Gould et al. [2009]), includes 715 colour images, of size 320 $\times$ 240 pixels, with ground-truth segmentation over 8 classes. A 5-fold cross validation is used with 572 images for the training, and 143 images for the test (Gould et al. [2009]). The above protocols split the datasets into predefined training-test sets.

4.3.2 Results & Discussion

The experimental outcomes are computed by comparing the outputs of classifiers and ground-truths; and are expressed as global and average precisions. The global accuracy is defined as the ratio of, correctly classified pixels, to the total number of pixels in the test set. On the other hand, average precision is calculated as the average of per-class accuracies, which are the ratio of, correctly classified pixels, to the total number of pixels in the same class.

Figure 4.2 illustrates the original, ground-truth, and colour-transformed images for MSRC-21 dataset. Tables 4.1, 4.2 and 4.3 report the quantitative results for MSRC-21 dataset on the TextonBoost framework and confirm that, the proposed colour transformation consistently outperforms other alternative subspace mappings, and colour spaces, with respect to the baseline. It is worth mentioning, that due to the fine tuning of parameters, preprocessing of images, or randomization functions, the accuracy of the TextonBoost implementation on different platforms is not consistent with that reported (Krähenbühl and Koltun [2012]).

Figure 4.3 shows original, correspondent segmentation, and the colour transformed samples. Table 4.4 presents per-class performances for Darwin. Tables 4.5 and 4.6 compare the results for different subspace mappings and colour transformations. It is clear that, the proposed colour transformation consistently improves both unary and pairwise classification accuracies, and outperforms other alternatives, including the baseline.

In summary, the class-driven colour transformation improves the performance of semantic segmentation for different datasets and frameworks; it outperforms other subspace mappings and standard colour transformations; and it is also consistent across both implementations.

Mapping	Building	Grass	Tree	Cow	Sheep	Sky	Airplane	Water	Face	Car	Bicycle	Flower	Sign	Bird	Book	Chair	Road	Cat	Dog	Body	Boat	Average	Global
Baseline	68	**98**	88	85	76	92	**86**	68	84	77	87	86	57	45	92	60	87	74	36	75	22	73.9	82.0
MMC	72	**98**	90	86	81	94	83	72	87	**84**	**89**	**93**	66	45	**97**	**68**	88	72	**42**	80	21	76.5	84.1
HDA	**73**	**98**	**91**	85	82	92	81	71	87	77	88	91	64	48	93	**68**	**89**	72	41	81	24	76.0	83.9
SVD	71	**98**	**91**	**87**	81	**95**	82	70	87	81	**89**	89	**70**	48	93	65	88	73	37	80	**30**	76.4	84.0
KPCA	72	**98**	89	85	**85**	94	**86**	70	87	82	88	90	66	**53**	96	56	88	**78**	38	80	24	76.3	83.9
Proposed	70	**98**	90	86	83	94	**86**	**74**	**89**	81	87	92	63	45	96	65	88	72	**42**	**84**	27	**76.8**	**84.2**

Table 4.1: Per-class accuracies for TextonBoost framework on MSRC-21 dataset. The combination of the projection with any of the backprojection schemes, outperforms the baseline, but metric MDS (Proposed) produces the highest precision.

Mapping	Unary		Pairwise	
	Average	Global	Average	Global
Baseline	67.1	78.9	70.2	82.8
MMC	68.1	78.5	71.3	82.7
HDA	67.4	77.9	70.8	82.2
SVD	67.3	79.8	69.1	83.5
KPCA	68.3	79.2	71.9	83.4
Proposed	**68.9**	**80.0**	**72.1**	**84.0**

Table 4.2: Precision of different subspace mappings for Darwin framework on MSRC-21 dataset. With the exception of KPCA and MDS (Proposed), other subspace mappings degrade the baseline performance with slight margins on both of the unary and pairwise accuracies.

Transformation	Unary		Pairwise	
	Average	Global	Average	Global
Baseline	67.1	78.9	70.2	82.8
YCrCb	65.4	77.8	69.5	81.8
HSL	65.2	77.0	70.3	82.1
Luv	65.9	78.1	70.3	82.1
I1I2I3	62.6	75.2	69.6	81.3
O1O2O3	62.3	75.1	68.9	81.2
Proposed	**68.9**	**80.0**	**72.1**	**84.0**

Table 4.3: Precision of different colour spaces for Darwin framework on MSRC-21 dataset. The baseline outperforms all standard colour spaces. However, the proposed colour transformation improves the accuracy, as it learns from data and class distributions.

Mapping	Sky	Tree	Road	Grass	Water	Building	Mountain	Foreground	Average	Global
Baseline	**86**	64	89	65	65	76	03	60	63.5	74.0
MMC	**86**	**68**	90	72	62	79	01	63	65.2	76.3
HDA	**86**	66	**91**	72	65	**80**	02	**65**	65.8	76.7
SVD	**86**	67	**91**	**73**	64	**80**	06	63	66.3	76.7
KPCA	**86**	**68**	**91**	**73**	65	**80**	03	61	65.9	76.6
Proposed	**86**	**68**	90	68	**67**	79	**09**	**65**	**66.5**	**76.8**

Table 4.4: Per-class accuracies for TextonBoost framework on SBD dataset. All the subspace mappings outperform the baseline, and MDS (Proposed) gives the best accuracy.

Mapping	Unary		Pairwise	
	Average	Global	Average	Global
Baseline	68.3	78.5	70.2	81.5
MMC	69.6	79.9	71.7	82.9
HDA	68.9	79.3	71.2	82.4
SVD	69.2	79.8	70.9	82.8
KPCA	68.5	78.8	70.7	82.0
Proposed	**70.4**	**81.4**	**72.5**	**84.2**

Table 4.5: Results of different subspace mappings for Darwin framework on SBD dataset. Both unary and pairwise precisions outperform the baselines for all the subspace mappings. MDS (Proposed) shows the best performance.

Transformation	Unary		Pairwise	
	Average	Global	Average	Global
Baseline	68.3	78.5	70.2	81.5
YCrCb	68.7	77.1	70.7	80.2
HSL	68.0	76.9	70.0	80.1
Luv	68.3	76.9	70.5	80.1
I1I2I3	67.1	75.9	68.6	78.7
O1O2O3	66.7	75.9	68.5	79.0
Proposed	**70.4**	**81.4**	**72.5**	**84.2**

Table 4.6: Results of different colour transformations for Darwin framework on SBD dataset. Only the YCrCb colour space shows slightly better unary and pairwise accuracies, when compared with the baseline. The proposed colour transformation gives the best results.

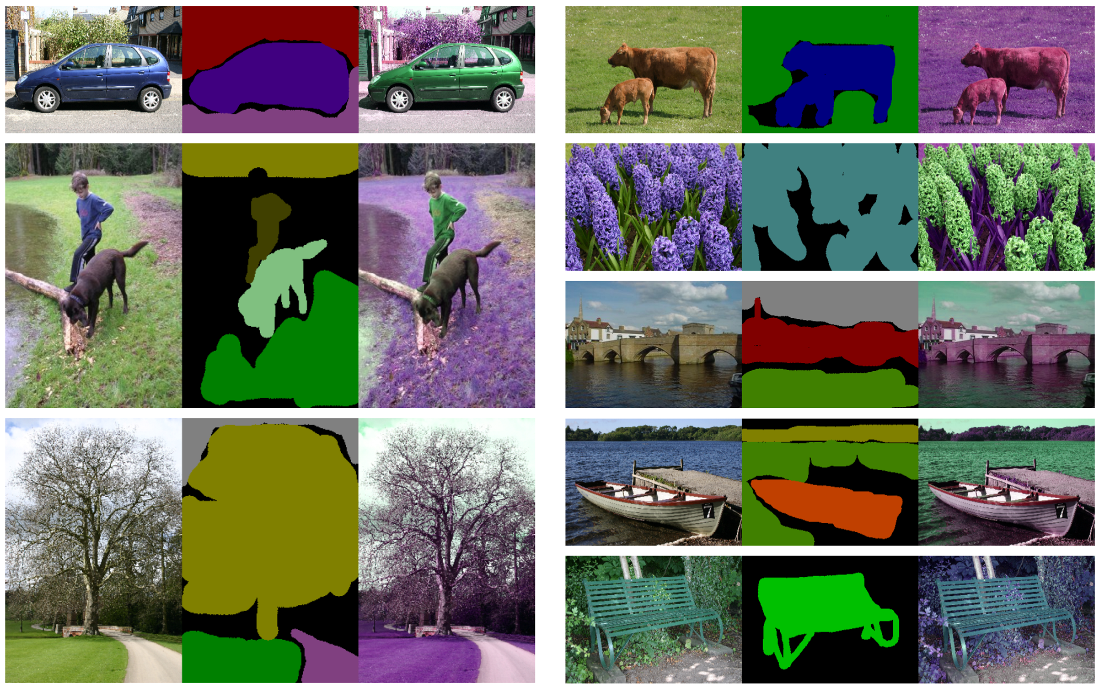
Figure 4.2: Sample images from the MSRC-21 dataset. Each panel shows raw RGB, corresponding segmentation, and transformed images.

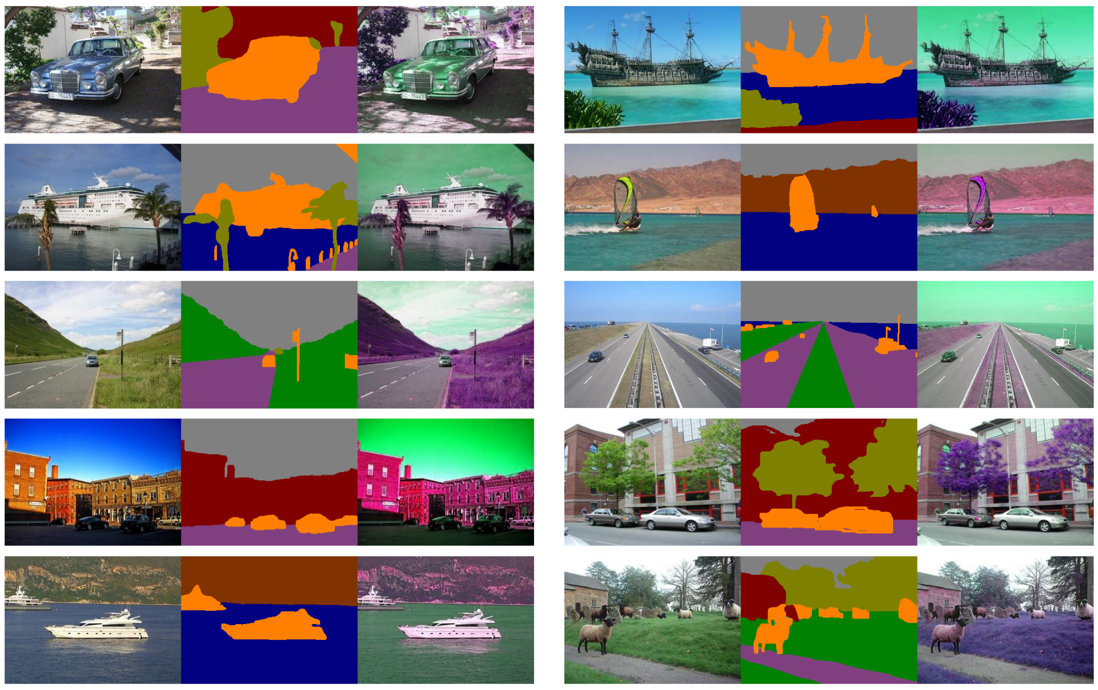
Figure 4.3: Sample images from the SBD dataset. Each panel shows raw RGB, corresponding segmentation, and transformed images.

4.4 Conclusion

In this chapter, the supervised colour transformation jointly utilized the colour transformation and supervised learning to accomplish semantic segmentation, *i.e.* pixelwise prediction of labels in a scene. This applied Fisher discrimination to expand the primary colour space into a high-dimensional space, with minimum distances among within-class pixels and maximum differences among between-class ones. After dimension expansion, a backprojection was proposed to reduce this high-dimensional space by metric MDS, which gave a new colour space with the same dimension as the primary space, while preserving the geodesics distances among classes in the high-dimensional space. This was also a departure from conventional linear dimension reduction, elsewhere in the literature. Finally, an anisotropic scaling-translation was employed to convert this colour space into a standard colour space. Experiments on two public databases, six canonical colour spaces, and three different subspace mappings confirmed that the proposed method outperformed other semantic segmentation algorithms.

Part II

Texture Classification

Chapter 5

Fisher Convolutional Neural Network

Texture understanding has always been a matter of interest to computer vision community. Texture is an important visual clue for classification and segmentation tasks, in various scene understanding challenges. A variety of methods are available for different tasks in texture understanding, for example, material recognition (Timofte and Van Gool [2012], Hu et al. [2011a]), texture perception (Cimpoi et al. [2015b]), and texture synthesis (Liu et al. [2012]).

The emergence of deep convolutional neural networks, has led to considerable improvements in image classification, detection and segmentation (Krizhevsky et al. [2012]), and particularly in texture recognition (Simonyan and Zisserman [2014]). Recently, deep filter banks proposed a new type of texture descriptor, computed by Fisher vector pooling of the convolutional networks. It resulted in significant improvements in the accuracy and performance of texture (Cimpoi et al. [2015a]), material (Badri et al. [2014]), and scene recognition (Oquab et al. [2014]).

5.1 Method

This chapter introduces a novel deep convolutional architecture, based on Fisher discrimination. It is specifically designed to learn scales, orientations and resolutions of texture filters. This network trains filter parameters, rather than the whole filters themselves, and gives a set of texture filters, which capture quality convolutional features in several layers of variable depths. This keeps the number of learning parameters, considerably lower than the classic networks. It is also able to learn numbers of texture filters independently and produce an ensemble of convolutional features, for the texture recognition.

The Fisher convolutional network can be trained in variable depth of feedforward neural layers, while the depth of layers for each texture filter, depends on its power to impose higher distinction into the convolutional features. This prevents redundant training for the texture filters, which are not capable of creating distinctive features in the deeper layers. The experiments show significant improvements in the precision for standard benchmarks.

5.2 Formulation

Assume a set of filters $\mathcal{F} = \{\mathbf{F}_1, \ldots, \mathbf{F}_{|\mathcal{F}|}\}$, such that, every sample filter $\mathbf{F} \in \mathcal{F}$ is generated with a specific function $\mathbf{f}(.)$, using three parameters *i.e.* scale (s), orientation (o), and resolution (r) as follows,

$$\mathbf{F} = \mathbf{f}(s, o, r) \tag{5.1}$$

The functions of set $\{\mathbf{f}_1(.), \ldots, \mathbf{f}_{|\mathcal{F}|}(.)\}$ are Gaussian or Laplacian of Gaussian (LoG). They generates texture filters, of size $r \times r$, to convolve with the input images, and extract texture features. The goal is to learn a set of optimal parameters for each filters of the set $\mathcal{F}$, such that, final ensemble of $|\mathcal{F}|$ convolutional features, improves the performance of texture classification task.

As a general preprocessing step, a set of texture images $\mathcal{I} = \{\mathbf{I}_1, \ldots, \mathbf{I}_{|\mathcal{I}|}\}$ are converted to standard CIE-Lab colour format and normalized to zero mean unit variance. This gives the chance of deploying all information in luminance and chrominance channels, for the purpose of texture understanding. The texture filter $\mathbf{F}$, with a set of initial parameter $\Theta = \{s, o, r\}$, is convolved $(*)$ with all the images in set $\mathcal{I}$, to obtain a set of filter outputs $\mathbf{R}$ as,

$$\mathbf{R}(\Theta) = \mathcal{I} * \mathbf{F} = \mathcal{I} * \mathbf{f}(\Theta) \tag{5.2}$$

Although it is possible to classify $\mathbf{R}$ directly, this can be projected to a space spanned by the number of classes in $\mathcal{I}$. It reflects the notion of texture classes, such that, separability between them gets maximized, whilst scattering within them, becomes minimized. This projection is presented as a matrix $\mathbf{A} \in \mathbb{R}^{d \times c}$ where d is the number of colour components for each pixel, and c stands for the number of texture classes.

The projection matrix $\mathbf{A}$ is multiplied by $\mathbf{R}$ to form a feature vector $\mathbf{V}$, such that,

$$\mathbf{V}(\Theta) = \mathbf{R}(\Theta)\mathbf{A} \tag{5.3}$$

The filter output $\mathbf{R}$ and feature vector $\mathbf{V}$, are nonlinearly related to the parameter set Θ through $\mathbf{f}(\Theta)$. This is in contrast with the notion of $\mathbf{F}$, which can be directly calculated by $\mathbf{f}(.)$ through plugging the parameters into the generator function.

After projection, the resulting feature vector $\mathbf{V}$ is employed to learn a K-Nearest Neighbour classifier (KNN), which is then evaluated by a validation set. It yields a quantitative measure of how each of the filter parameters of set Θ contribute to the texture recognition performance. This recognition performance is maximized by updating these filter parameters. Here, every performance measures, such as ϕ, derived by confusion matrix $\mathbf{C}$ of the KNN classifier, can be minimized to update the filter parameters. For this optimization problem, a convex objective function is defined as follows,

$$\mathcal{P}(\Theta) = -log(\phi) \tag{5.4}$$

This is a nonlinear function, with respect to the parameter set Θ and feature vector $\mathbf{V}$. The minimization of $\mathcal{P}$ by a nonlinear optimization technique, gives the optimal set of parameters Θ^*. This generates the optimized texture filter $\mathbf{F}^*$, that lets to compute the optimal projection matrix $\mathbf{A}^*$. Figure 5.1 illustrates a learning neuron of the Fisher convolutional neural network. This process begins by the initial set of parameters Θ and Equation 5.2 is employed to minimize Equation 5.4. After ith iteration, it gives $\Theta^{(i)}$ to calculate $\mathbf{F}^{(i)}$. The feature vector $\mathbf{V}^{(i)}$ is then classified and the confusion matrix $\mathbf{C}^{(i)}$ is drawn. If the recognition performance improves well and passes a predefined threshold, the optimal filter $\mathbf{F}^*$ is set to $\mathbf{F}^{(i)}$ and an activation function is applied to feed the input of the next neuron. Otherwise, the ith set of filter parameters, is considered as the initial point to run the $(i+1)$th iteration, that generates the texture filter $\mathbf{F}^{(i+1)}$ following the same procedure.

Recent developments in deep learning, prove the likelihood of drawing better high-order features, by deep multilayer neural networks. Hence, the above process can move forward in consecutive convolutional layers, applying Fisher discrimination in a layer-wise manner. This continues until no meaningful improvement is gained in the performance.

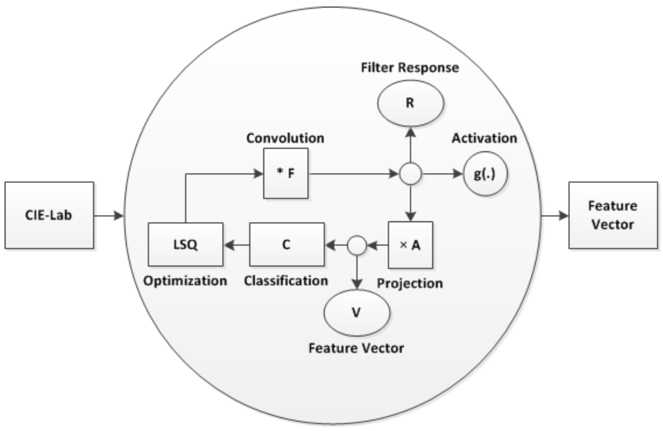

Figure 5.1: Learning neuron of Fisher convolutional neural network. The input is convolved by the initial filter, is projected to a space spanned by the number of input classes, and then is classified. This process continues by updating the filter parameters until no further improvement in classification precision can be made. The convolutional features (response of filters with optimized parameters) pass through an activation function to feed the next neuron.

Following the successful practice of applying activation functions for transferring of features between deep layers, $g(x) = \frac{x}{1+|x|}$ (Softsign) is employed in the output of each neurons. This activation is robust to the initialization and shows gentle nonlinearity (Glorot and Bengio [2010]). It also preserves separability imposed by the projection, and avoids vanishing of high-order gradients.

In the Fisher convolutional neural network, each texture filter of the set $\mathcal{F}$ is trained separately and hence, $|\mathcal{F}|$ set of optimal parameters are produced. For texture recognition, the above optimal filters are applied on the test images, to produce an ensemble of convolutional features. Following the successful practice of deep filter banks (Cimpoi et al. [2015a]), dictionary learning is employed to feed a SVM to classify the dictionary of codes. Figure 5.2 depicts an example of Fisher convolutional network.

5.2.1 Projection

Formerly, a projection matrix $\mathbf{A} \in \mathbb{R}^{d \times c}$ is defined, that maps d colour components into a space corresponding to the $c > d$ texture classes. Suppose that $\mathbf{R}$ contains the filter responses of n pixels in c texture classes of the set $\mathcal{I}$.

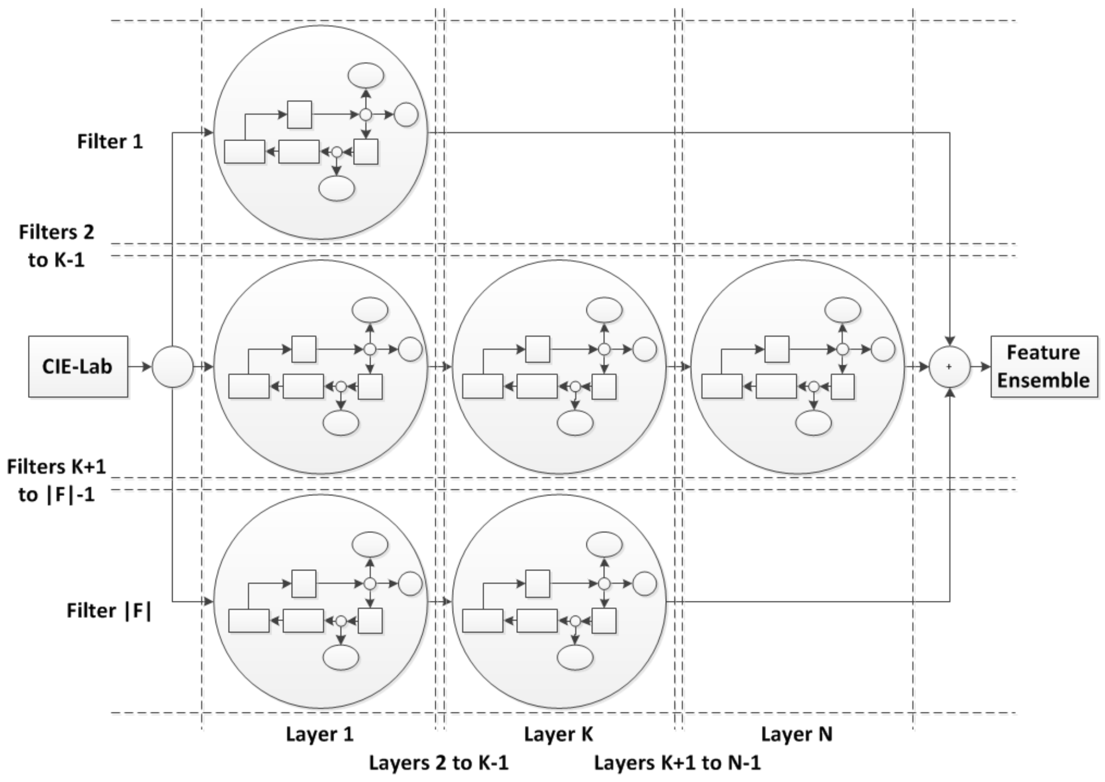

Figure 5.2: Fisher convolutional neural network. Each convolutional filters is trained by a specific layer, consisting of several learning neurons. The depths of distributed layers, varies with respect to the capability of their corresponding filters to extract more distinct features. As a result, the final ensemble of features, has great discrepancy among classes of the input. This leads to better classification performance by any classifier of choice.

Inspired by the concept of Fisher's criterion (Bishop [2006]), the aim is to minimize the ratio of inter-intra class scatterings of set $\mathcal{S}_{\mathcal{A}} = \{\mathbf{S_{wA}}, \mathbf{S_{bA}}\}$ by figuring out $\mathbf{A}$, such that,

$$\arg\min_{\mathbf{A}\mathbf{A}^T=\mathbf{I}} \mathcal{H}(\mathbf{A}) = tr\left[(\mathbf{A}\mathbf{S_{wA}}\mathbf{A^T})(\mathbf{A}\mathbf{S_{bA}}\mathbf{A^T})^{-1}\right] \tag{5.5}$$

Here, $tr(.)$ is the trace operator, $\mathbf{I}$ is the identity matrix, and $\mathbf{AA^T} = \mathbf{I}$, imposes orthogonality into the projection matrix $\mathbf{A}$.

Since the CIE-Lab colour space is used, the filter outputs $\mathbf{R}$ belong to $\mathbb{R}^{n\times d}$ where $d = 3$. It is also required that $\mathbf{A}$ belongs to $\mathbb{R}^{d\times c}$ to hold $\mathbf{V}$ in $\mathbb{R}^{n\times c}$ and make Equation 5.3 consistent in its dimensions. This also forces the set $\mathcal{S}_{\mathcal{A}}$ to stand in $\mathbb{R}^{c\times c}$ for making Equation 5.5, dimensionally consistent.

This means that, it is impossible to employ classic discriminant analysis (Bishop [2006]) to solve Equation 5.5, because $c > d$ and hence, this is no longer a dimension reduction problem. Note that the dimension of target space **V**, is higher than the original space **R**, for all available texture datasets. To handle this issue, the scattering set $\mathcal{S}_{\mathcal{A}}$ is redefined with respect to the dimensionality of target space (c) rather than the primary CIE-Lab colour space (d).

For $\mathbf{S_{wA}}$, consider a square matrix, of size $c \times c$, with zero in all entries except main diagonal. The *j*th entry of $\mathbf{S_{wA}}$ is calculated as follows,

$$\mathbf{S_{wA}}(j,j) = tr\Big[(x_j - \mu_j)(x_j - \mu_j)^T\Big] \quad \forall\, j \in [1,c] \tag{5.6}$$

The x_j and μ_j are convolutional features and average vectors over *j*th texture class of **R**. The $\mathbf{S_{bA}}$ can also be defined as another square matrix, of size $c \times c$, that all of its entries are set to zero, except main diagonals,

$$\mathbf{S_{bA}}(j,j) = tr\Big[(\mu_j - \bar{\mu})(\mu_j - \bar{\mu})^T\Big] \quad \forall\, j \in [1,c] \tag{5.7}$$

where $\bar{\mu}$ is the average vector over all texture classes. This is aligned with the objective function in Equation 5.5 which tries to capture the separation power through trace operator.

To solve Equation 5.5, Fast Iterative Shrinkage Thresholding Algorithm (FISTA) (Beck and Teboulle [2009]), is employed, as a gradient descent method with proven fast convergence. The optimization begins with random initialization of $\mathbf{A}^{(0)}$ to determine the optimal projection matrix **A**, for each texture filter of the bank $\mathcal{F}$. For the purpose of implementation, UnLocBox toolbox (Combettes and Pesquet [2011]), is utilized.

5.2.2 Optimization

After projecting the filter output **R** by the matrix **A**, to produce feature vector **V**, it is necessary to deploy a proper optimization algorithm to minimize Equation 5.4 as follows,

$$\underset{\Theta}{\operatorname{argmin}}\ \mathcal{P} = -log\Big[\phi\big(\mathbf{V}(\Theta)\big)\Big] \tag{5.8}$$

Here, ϕ is the performance measure derived by the confusion matrix $\mathbf{C}$, generated by the KNN classifier inside the neuron. Instead of solving $\mathcal{P}$, its counterpart $\mathcal{Q}$ can be solved with less efforts. Lets define $\mathcal{Q}$ as a least-squares minimization problem,

$$\underset{\Theta}{\operatorname{argmin}}\ \mathcal{Q} = \Big[log\big(\rho(\Theta)\big)\Big]^2 + \Big[log\big(\eta(\Theta)\big)\Big]^2 + \Big[log\big(\chi(\Theta)\big)\Big]^2 \tag{5.9}$$

The real functions ρ, η and χ are global, average and recall precisions, derived from the confusion matrix $\mathbf{C}$. By definition, ρ is the proportion of the total number of predictions that are correct, η is a measure of the accuracy provided that a specific class has been predicted, and χ is a measure of the ability of a prediction model to select instances of a certain class from a dataset (Sammut and Webb [2011]). In practice, negative logarithms of precisions, is considered as auxiliary functions, which optimize $\mathcal{P}$ through minimizing of $\mathcal{Q}$. The goal is to maximize the classification performance by imposing symmetry to the confusion matrix $\mathbf{C}$ to avoid biases towards majority or minority texture classes.

To solve the Equation 5.9, nonlinear least-squares minimization with trust-region-reflective (Coleman and Li [1996]), is implemented by the built-in function of Matlab optimization toolbox. As a rule of thumb, the number of neighbours in KNN classifiers are set to the square root of feature vector length.

5.3 Experiments

For the experiments, three well-known texture filter banks (LM, MR and Schmid) are employed. The Fisher convolutional neural network is employed to learn their optimal parameters for five texture datasets (UIUC, KTH-TIPS2-a, KTH-TIPS2-b, FMD and DTD). The code is embedded in Oxford Visual Geometry Group's implementation and the mean accuracy of recognition, averaged over standard number of splits, is reported according to a standard evaluation protocol (Cimpoi et al. [2014a]).

This follows the well-known trend of computing local image descriptors and encoding them into a visual dictionary. The current descriptors are 128-dimensional dense SIFT features (DSIFT) computed for bins, of size 6×6 pixels, at scales $\{1, \frac{\sqrt{2}}{2}, \frac{1}{2}, \frac{\sqrt{2}}{4}, \frac{1}{4}\}$.

The descriptors are soft quantized by Gaussian Mixture Model (GMM) and normalized to generate Improved Fisher Vectors (IFV). After normalization, a linear SVM classifier is trained, and the validation set is used to find its regularization parameter. In each experiments, other deep descriptors are replaced with DSIFT in the standard pipeline of texture recognition.

5.3.1 Filter Banks

The evaluations consider three filter banks, each consisting of 99 filters, of size 49×49 pixels, divided into 10 categories, according to their generation functions. The first bank is Leung-Malik (LM) (Leung and Malik [2001]), including 36 first and second derivatives of the Gaussian filters, at three scales $\{\sqrt{2}, 2, 2\sqrt{2}\}$ and six orientations $\{\frac{\pi}{6}, \frac{\pi}{3} \ldots, \pi\}$, eight LoG, and four Gaussian filters at scales $\{\sqrt{2}, 2, 2\sqrt{2}, 4\}$. The second bank is Maximum-Response (MR) (Varma and Zisserman [2003]), consisting of 36 filters at three scales $\{1, 2, 4\}$ and six orientations added to two isotropic Gaussian and LoG filters. The third bank is Schmid (S) (Schmid [2001]), containing 13 rotationally invariant filters with $\sigma \in \{2, 4, 6, 8, 10\}$ and $\tau \in \{1, 2, 3, 4\}$.

5.3.2 Datasets

The UIUC texture database (Lazebnik et al. [2005]), contains 1000 images (40 samples, 25 classes) in grayscale format. KTH-TIPS2-a and KTH-TIPS2-b (Mallikarjuna et al. [2006], Timofte and Van Gool [2012]), stand for Textures under varying Illumination, Pose and Scale. The former uses only 72 images for 4 out of 44 samples, while the latter consists of 4572 images (4 samples, 108 images per sample and 11 categories). The Flicker Material Dataset (FMD) (Sharan et al. [2009]), includes 1000 samples (100 images per category, 10 categories), selected manually from Flickr (Sharan et al. [2009]). The Describable Texture Dataset (DTD) (Cimpoi et al. [2014b]), contains 5640 annotated texture images, with one or more adjectives from 47 English words (120 representative images per attribute), and 10 pre-set splits into training-validation-test sets.

Dataset	DSIFT	DeCAF	VGG	Fisher
UIUC	96.6±0.7	96.4±0.7	96.7±1.5	**97.3±0.5**
KTH-a	68.4±5.6	73.9±4.2	73.3±3.9	**77.6±3.6**
KTH-b	69.8±5.7	74.8±4.3	73.2±3.9	**75.1±3.1**
FMD	50.3±1.4	63.2±1.4	65.5±1.0	**69.6±1.5**
DTD	59.6±1.0	57.6±1.5	58.3±1.1	**63.3±0.9**

Table 5.1: Mean accuracy of texture recognition by dictionary learning for dense SIFT (DSIFT), DeCAF(FC6), VGG-VD and Fisher convolutional features. It can be seen that, the proposed Fisher convolutional features create better dictionaries by imposing higher discrepancies among texture classes, to the advantage of the classifier.

5.3.3 Results & Discussion

Table 5.1 presents the performances in terms of mean accuracy for the texture recognition. The results are reported to compare with recent state-of-the-arts in the literature (Cimpoi et al. [2015a]). The first column represents precisions, coming from the dense SIFT (DSITF), followed by DeCAF(FC6) (Donahue et al. [2013]), VGG-VD (Simonyan and Zisserman [2014]), and the proposed Fisher convolutional features.

It is obvious that the Fisher convolutional features perform better than DSIFT on all the datasets under examination. For FMD dataset, it shows a two-digit margin for precision improvement. This effect is also observed for DeCAF and there are significant improvements on FMD and DTD datasets. Compared to VGG-VD, outcomes are close with a small margin on UIUC dataset and great improvements on the rest of experimental datasets.

Figure 5.3 illustrates the initial and corresponding learned filters. Some texture filters remain unchanged, because their initial parameters are already optimum for the texture recognition task. For other filters, either all or some of their parameters are updated, leading to variations in their scales, rotations or resolutions. As a result, they represent greatly different shapes, compared to the initial texture filters.

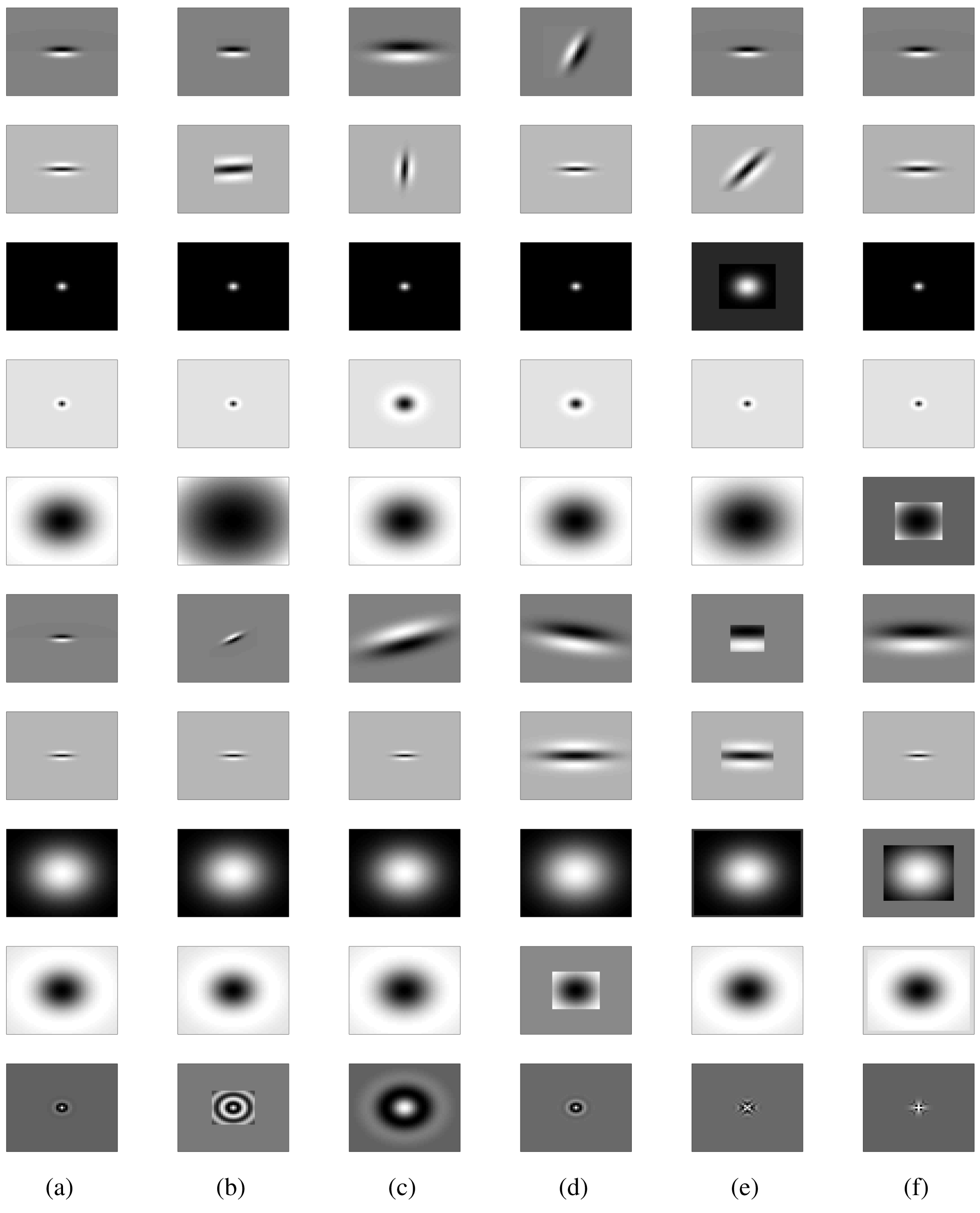

Figure 5.3: Examples of (a) initial filters and their corresponding texture filters, learned for the (b) UIUC; (c) KTH-TIPS2-a; (d) KTH-TIPS2-b; (e) FMD; (f) DTD datasets. The variations in scales, orientations and resolutions of the trained filters are tailored to the datasets, that these filters learn from. For directional filters, the orientations are prone to considerable change; whereas, for rotational filters, resolutions are the main source of variations.

5.4 Conclusion

In this chapter, an architecture incorporating Fisher discrimination with convolutional neural network, was proposed to perform texture classification by searching for a set of filters which yielded convolutional features with the best classification performances. Each filter was determined by its scale, orientation and resolution. The Fisher discrimination was employed to create highly separated features which were subsequently used to construct a KNN classifier, that its performance was optimized in consequent iterations over above parameters. The number of iterations with a filter formed a learning neuron, which its output passed through an activation function and served as the input of next layer, to calculate high-order features. Dictionary learning and SVM were deployed to classify the final set of features that in extensive experiments, showed better classification performance over the recent baselines.

Chapter 6

Fisher Convolutional Autoencoder

Learning of convolutional filters in deep neural networks, provides sparse representations for the purpose of image recognition. Autoencoders are a family of powerful deep neural networks, used to build scalable, generative models for automatic feature learning. They introduce a powerful variation in the way that, hierarchical features are learned. An autoencoder is a neural network, trained to prioritize useful aspects of the input data. Traditionally, autoencoders were used for dimensionality reduction or feature learning, but recently, their theoretical connections with latent variable models have brought them to the forefront of generative modelling (Goodfellow et al. [2016b]).

A regularized autoencoder learns the most salient features of the data distribution by limiting model capacity. To this end, this keeps the autoencoder shallow and the code size small. The model has other properties, such as, sparsity, smallness of the derivative of the representation, and robustness to noise or missing inputs. In contrast, a variational autoencoder (Kingma et al. [2014]) and generative stochastic networks (Bengio et al. [2013]), learn high-capacity and overcomplete encodings of the input, without regularization. Besides, a convolutional autoencoder network is trained, using online gradient descent without additional regularization terms, and is properly scaled to the high-dimensional inputs (Masci et al. [2011]). Figure 6.1 shows the architecture of an overcomplete autoencoder.

6.1 Method

A Fisher convolutional autoencoder network is inspired by the stacked hierarchy of autoencoders, and is able to learn parametric and separable texture filters, in a novel deep architecture. Figure 6.2 represents the architecture of the Fisher convolutional autoencoder.

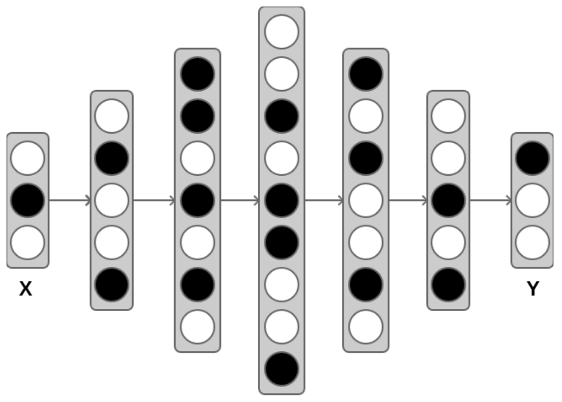

Figure 6.1: Overcomplete autoencoder. This is a stacked regularized autoencoder, trying to reconstruct noisy inputs (X) based on stacking layers, trained locally to remove noise from the corrupted versions of the input, to form the output (Y).

This overcomplete autoencoder, employs the Fisher discrimination to impose the highest possible distinction among texture classes. It does so while holding the minimum separation within each classes of the dataset under study. A network of Fisher convolutional autoencoders, learns banks of texture filters to makes an ensemble of deep convolutional features, which benefits from higher separability, and enables better classification. This network automatically adjusts the depth of each stack with respect to the capability of its corresponding texture filter, upon extracting high distinction features. Each stack of the Fisher convolutional autoencoders, is trained for a specific texture filter, using distributed Fisher discrimination (Section 3.3).

Figure 6.3 depicts a Fisher convolutional autoencoder network. The experiments are conducted on several publicly available datasets, which vary in the number of classes and the quality of texture samples, using a standard implementation. The results confirm the supremacy of this deep architecture to improve the precision of texture classification.

6.2 Formulation

The Fisher convolutional autoencoder operates by a projection-backprojection formulation, derived by Fisher discrimination. This formulation replaces coding-decoding part of a conventional overcomplete autoencoder. Instead of reconstructing the noisy input, it aims to impose the highest class separability in the output, with respect to the input.

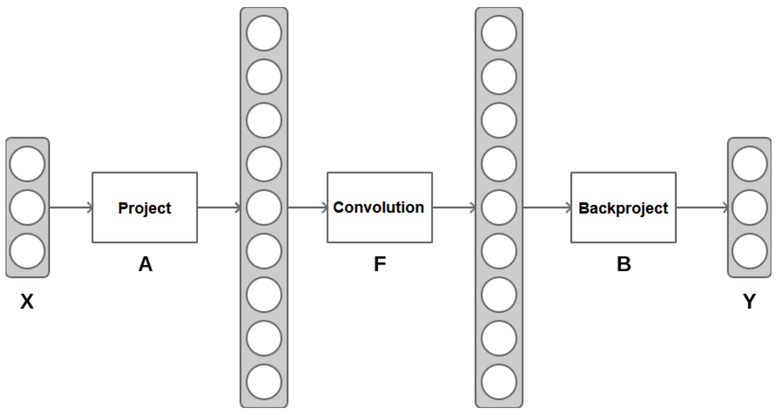

Figure 6.2: Fisher convolutional autoencoder. This consists of a projection to higher dimensions, followed by a convolution and then, a backprojection to the primary dimensions. It aims to expose better separation between classes of output (Y) than input (X).

For projection, a mapping matrix $\mathbf{A} \in \mathbb{R}^{d\times c}$ minimizes the ratio of between-within class scatterings of set $\mathcal{S}_{\mathcal{A}} = \{\mathbf{S}_{\mathbf{w}A}, \mathbf{S}_{\mathbf{b}A}\}$, by imposing orthogonality,

$$\underset{\mathbf{A}}{\operatorname{argmin}}\ \mathcal{Q}(\mathbf{A}) = tr(\mathbf{A}\mathbf{S}_{\mathbf{w}A}\mathbf{A}^{\mathbf{T}})\left[tr(\mathbf{A}\mathbf{S}_{\mathbf{b}A}\mathbf{A}^{\mathbf{T}})\right]^{-1} + \lambda_1\|\mathbf{I} - \mathbf{A}\mathbf{A}^{\mathbf{T}}\|_2 \tag{6.1}$$

The first term of $\mathcal{Q}(\mathbf{A})$ corresponds to the Fisher's criterion (quotient of trace) which aims to make the highest possible separability among texture classes. The second term is a regularizer, imposing orthogonality into the projection matrix $\mathbf{A}$. Looking back at Section 3.2, one can see that the set of eigenvectors corresponding to the largest c eigenvalues of $\mathbf{S}_{\mathbf{w}A}^{\mathbf{-1}}\mathbf{S}_{\mathbf{b}A}$ is a solution for this equation. Here, it is taken as an initial projection matrix $\mathbf{A}^{(\mathbf{0})}$ to solve Equation 6.1. This is due to the fact that, the classic fisher criterion is defined as trace-of-quotient, which can be solved by generalized eigenvalue method but, $\mathcal{Q}(\mathbf{A})$ is formed as the quotient-of-trace (Cunningham and Ghahramani [2015]).

Following the same practice as projection, the minimization problem for a backprojection matrix $\mathbf{B} \in \mathbb{R}^{c\times d}$ is defined by,

$$\underset{\mathbf{B}}{\operatorname{argmin}}\ \mathcal{P}(\mathbf{B}) = tr\big(\mathbf{B}^{\mathbf{T}}\mathbf{S}_{\mathbf{w}B}\mathbf{B}\big)\left[tr\big(\mathbf{B}^{\mathbf{T}}\mathbf{S}_{\mathbf{b}B}\mathbf{B}\big)\right]^{-1} + \lambda_2\|\mathbf{I} - \mathbf{B}^{\mathbf{T}}\mathbf{B}\|_2 \tag{6.2}$$

where $\mathbf{B}^{(\mathbf{0})}$ is set to the largest d eigenvalues of $\mathbf{S}_{\mathbf{w}B}^{\mathbf{-1}}\mathbf{S}_{\mathbf{b}B}$. Closed form formulations of gradients for Equations 6.1 and 6.2 can be worked out as follows.

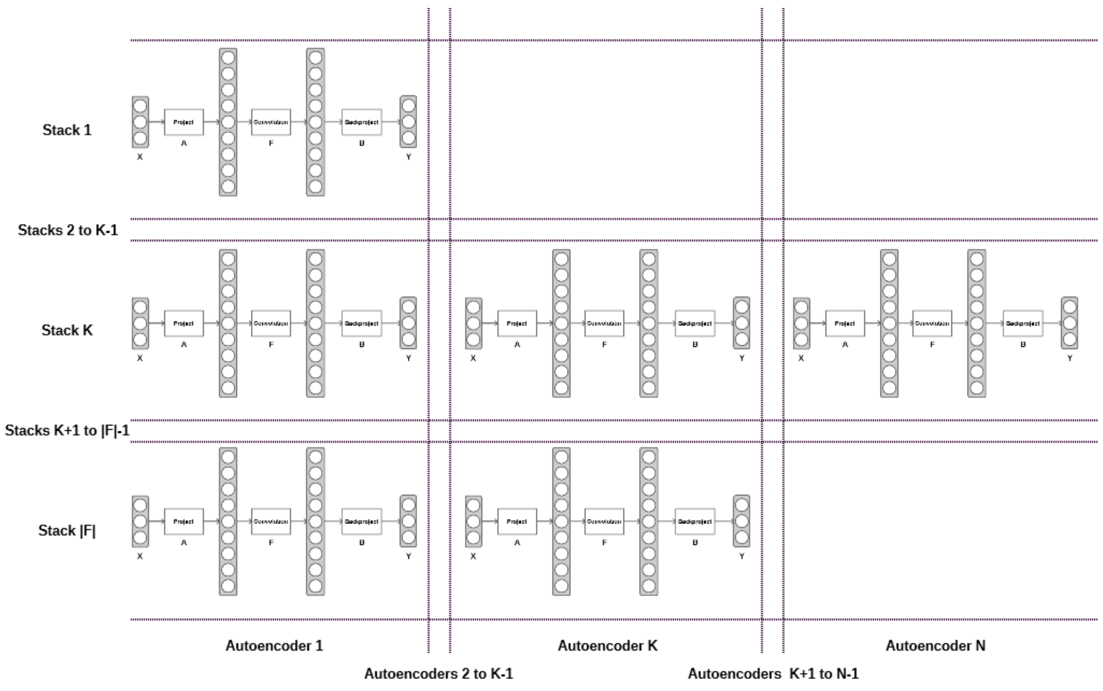

Figure 6.3: Fisher convolutional autoencoder network. This includes $|F|$ stacks with maximum N Fisher convolutional autoencoders, arranged at different depths, depending upon each filter's ability to better distinguish between classes. Dictionary learning is employed for each of the parallel stacks, and dictionary codes are ensembled to feed a classifier of choice.

Suppose that $\mathcal{Q}(\mathbf{A})$ is composed of $\mathcal{Q}_1(\mathbf{A})$ and $\mathcal{Q}_2(\mathbf{A})$, such that,

$$\mathcal{Q}_1(\mathbf{A}) = tr\left(\mathbf{A}\mathbf{S}_{\mathbf{w}A}\mathbf{A}^{\mathbf{T}}\right)\left[tr\left(\mathbf{A}\mathbf{S}_{\mathbf{b}A}\mathbf{A}^{\mathbf{T}}\right)\right]^{-1} \tag{6.3}$$

$$\mathcal{Q}_2(\mathbf{A}) = \|\mathbf{I} - \mathbf{A}\ \mathbf{A}^{\mathbf{T}}\|_2 \tag{6.4}$$

According to matrix calculus (Petersen et al. [2008]),

$$\frac{\partial tr\left(\mathbf{A}\mathbf{S}_{\mathbf{w}A}\mathbf{A}^{\mathbf{T}}\right)}{\partial \mathbf{A}} = \left(\mathbf{S}_{\mathbf{w}A}^{\mathbf{T}} + \mathbf{S}_{\mathbf{w}A}\right)\mathbf{A}^{\mathbf{T}} \tag{6.5}$$

$$\frac{\partial tr\left(\mathbf{A}\mathbf{S}_{\mathbf{b}A}\mathbf{A}^{\mathbf{T}}\right)}{\partial \mathbf{A}} = \left(\mathbf{S}_{\mathbf{b}A}^{\mathbf{T}} + \mathbf{S}_{\mathbf{b}A}\right)\mathbf{A}^{\mathbf{T}} \tag{6.6}$$

Algorithm 2 Supervised Projection

```
**Input:** $\mathcal{X} \in \mathbb{R}^{n \times d}$ ($n$ pixels of depth $d$)
**Output:** projection matrix $\mathbf{A} \in \mathbb{R}^{d \times c}$

**for** $k = 1$ **to** $|\mathcal{F}|$ **do**
  1: Compute $\mathbf{S_{wA}}$ (Equation 3.16), $\mathbf{\Gamma}_w$ (Equation 3.19)
  2: Calculate $\mathbf{S_{bA}}$ (Equation 3.24)
  3: Set $\mathbf{A}^{(0)}$ as $c$ largest eigenvalues of $\mathbf{S_{wA}^{-1} S_{bA}}$
  4: Solve Equation 6.1 by using Equation 6.10
**end for**
```

and hence,

$$\begin{aligned}\frac{\partial \mathcal{Q}_1}{\partial \mathbf{A}} &= \frac{tr\left(\mathbf{A}\mathbf{S}_{\mathbf{w}A}\mathbf{A}^{\mathbf{T}}\right)}{\left[tr\left(\mathbf{A}\mathbf{S}_{\mathbf{b}A}\mathbf{A}^{\mathbf{T}}\right)\right]^2}\left(\mathbf{S}_{\mathbf{b}A}^{\mathbf{T}} + \mathbf{S}_{\mathbf{b}A}\right)\mathbf{A}^{\mathbf{T}} \\ &- \frac{1}{tr\left(\mathbf{A}\mathbf{S}_{\mathbf{b}A}\mathbf{A}^{\mathbf{T}}\right)}\left(\mathbf{S}_{\mathbf{w}A}^{\mathbf{T}} + \mathbf{S}_{\mathbf{w}A}\right)\mathbf{A}^{\mathbf{T}} \qquad (6.7)\end{aligned}$$

On the other hand,

$$\frac{\partial \mathcal{Q}_2}{\partial \mathbf{A}} = \left(\frac{\partial\left(\mathbf{I} - \mathbf{A}\mathbf{A}^{\mathbf{T}}\right)}{\partial \mathbf{A}}\right)\frac{\mathbf{I} - \mathbf{A}\mathbf{A}^{\mathbf{T}}}{\|\mathbf{I} - \mathbf{A}\mathbf{A}^{\mathbf{T}}\|_2} \qquad (6.8)$$

which gives,

$$\frac{\partial \mathcal{Q}_2}{\partial \mathbf{A}} = \frac{-2\mathbf{A}^{\mathbf{T}}\left(\mathbf{I} - \mathbf{A}\mathbf{A}^{\mathbf{T}}\right)}{\|\mathbf{I} - \mathbf{A}\mathbf{A}^{\mathbf{T}}\|_2} \qquad (6.9)$$

and finally,

$$\begin{aligned}\frac{\partial \mathcal{Q}}{\partial \mathbf{A}} &= \frac{tr\left(\mathbf{A}\mathbf{S}_{\mathbf{w}A}\mathbf{A}^{\mathbf{T}}\right)}{\left[tr\left(\mathbf{A}\mathbf{S}_{\mathbf{b}A}\mathbf{A}^{\mathbf{T}}\right)\right]^2}\left(\mathbf{S}_{\mathbf{b}A}^{\mathbf{T}} + \mathbf{S}_{\mathbf{b}A}\right)\mathbf{A}^{\mathbf{T}} \\ &- \frac{1}{tr\left(\mathbf{A}\mathbf{S}_{\mathbf{b}A}\mathbf{A}^{\mathbf{T}}\right)}\left(\mathbf{S}_{\mathbf{w}A}^{\mathbf{T}} + \mathbf{S}_{\mathbf{w}A}\right)\mathbf{A}^{\mathbf{T}} \\ &- \frac{2\lambda_1}{\|\mathbf{I} - \mathbf{A}\mathbf{A}^{\mathbf{T}}\|_2}\mathbf{A}^{\mathbf{T}}\left(\mathbf{I} - \mathbf{A}\mathbf{A}^{\mathbf{T}}\right) \qquad (6.10)\end{aligned}$$

Algorithm 2 summarizes the computing of **A**, through supervised projection.

Algorithm 3 Supervised Backprojection

```
Input: $\mathcal{Y} = \mathcal{X}\mathbf{A} * \mathcal{F} \in \mathbb{R}^{n \times c}$ ($n$ pixels of depth $c$)
Output: backprojection matrix $\mathbf{B} \in \mathbb{R}^{c \times d}$

for $k = 1$ to $|\mathcal{F}|$ do
    1: Compute $\mathbf{S_{wB}}$ (Equation 3.2) and $\mathbf{S_{bB}}$ (Equation 3.3)
    2: Set $\mathbf{B}^{(0)}$ as $d$ largest eigenvalues of $\mathbf{S_{wB}^{-1} S_{bB}}$
    3: Solve Equation 6.2 by using Equation 6.11
end for
```

The backprojection algorithm is a subspace mapping, providing $\mathbf{B} \in \mathbb{R}^{c \times d}$, to reduce the dimension of convolutional features from c to the initial dimension d by,

$$\begin{aligned}
\frac{\partial \mathcal{P}}{\partial \mathbf{B}} &= \frac{tr\big(\mathbf{B^T S}_{\mathbf{w}B}\mathbf{B}\big)}{\big[tr\big(\mathbf{B^T S}_{\mathbf{b}B}\mathbf{B}\big)\big]^2}\big(\mathbf{S}_{\mathbf{b}B} + \mathbf{S}_{\mathbf{b}B}^{\mathbf{T}}\big)\mathbf{B} \\
&- \frac{1}{tr\big(\mathbf{B^T S}_{\mathbf{b}B}\mathbf{B}\big)}\big(\mathbf{S}_{\mathbf{w}B} + \mathbf{S}_{\mathbf{w}B}^{\mathbf{T}}\big)\mathbf{B} \\
&- \frac{2\lambda_2}{\|\mathbf{I} - \mathbf{B^T B}\|_2}\mathbf{B}\big(\mathbf{I} - \mathbf{B^T B}\big) \qquad (6.11)
\end{aligned}$$

Algorithm 3 presents the computing of backprojection matrix **B** by supervised backprojection. For implementation, FISTA (Beck and Teboulle [2009]) is employed.

6.3 Experiments

The proposed network is employed to learn scales, orientations and resolutions of three well-known texture descriptors (LM, MR, Schmid) for five publicly-available texture datasets (UIUC, KTH-TIPS2-a, KTH-TIPS2-b, FMD, DTD). The implementation is embedded in the Oxford Visual Geometry Group's platform for texture understanding (Cimpoi et al. [2014b]). According to the evaluation protocols, mean accuracy of recognition, averaged over standard number of splits, is reported for each experiments. These include dictionary learning of features, generated by the Fisher convolutional autoencoder network (Fisher), and the state-of-the-arts in literature, *i.e.* dense SIFT (DSIFT) (Chatfield et al. [2014], DeCAF(FC6) (Donahue et al. [2013]), and VGG-VD (Simonyan and Zisserman [2014]).

6.3.1 Parametric Texture Filters

Suppose a set of Gaussian or Laplacian of Gaussian (LoG) filters $\mathcal{F} = \{\mathbf{F}_1, \ldots, \mathbf{F}_{|\mathcal{F}|}\}$. For each individual filter $\mathbf{F} \in \mathcal{F}$, the input $\mathcal{X}$ is projected by the matrix $\mathbf{A}$. This creates a latent projected vector, that is convolved with the filter $\mathbf{F}$ to generate a latent convolutional feature set. This set backprojects by matrix $\mathbf{B}$ to the output $\mathcal{Y}$. The aim is to maximize separation, and minimize scattering among texture classes.

The solution is a set of optimal filter parameters Θ^* that finally, provides the optimal texture filter $\mathbf{F}^*$ for the convolutional autoencoder. In this instance, the Matlab optimization toolbox (Coleman and Li [1996]), is used to implement the nonlinear least-squares minimization with trust-region-reflective algorithm.

Table 6.1 presents results of texture classification experiments. It can be seen that the performance of the Fisher convolutional autoencoder network (Fisher) outperforms on FMD and KTH-TIPS2-a datasets. Results for VGG-VD is better than the proposed network on KTH-TIPS2-b and DTD datasets, but on UIUC dataset, the proposed algorithm follows DSIFT. This is due to the fact that, UIUC is a grayscale texture dataset and hence, the proposed projection-backprojection paradigm cannot take advantage of orthogonality in the standard colour spaces, to impose powerful separation among the texture classes.

Table 6.2 shows mean accuracy for the concatenation of the Fisher convolutional features, with DeCAF deep descriptors. The model, pretrained on ImageNet (Donahue et al. [2013]), is employed to compute DeCAF features for all datasets except DTD dataset, whose its features are available on the web. The results (Fisher + DeCAF) are reported besides DSIFT + DeCAF (Cimpoi et al. [2014b]). It is obvious that the proposed network outperforms on all datasets, with considerable improvements in the recognition performance.

Table 6.3 provides the outcomes of the same experiment for VGG-VD features, pretrained on ImageNet (Simonyan and Zisserman [2014]). According to these results, the performances of Fisher + VGG on all datasets, are improved against DSIFT + VGG (Cimpoi et al. [2015b]).

Dataset	DSIFT	DeCAF	VGG	Fisher
UIUC	96.6±0.7	96.4±0.7	96.7±1.5	**97.2±1.3**
KTH-a	68.4±5.6	**73.9±4.2**	73.3±3.9	72.1±3.6
KTH-b	69.0±5.7	**74.8±4.3**	73.2±3.9	71.8±4.5
FMD	50.3±1.4	63.2±1.4	**65.5±1.0**	52.5±3.2
DTD	59.6±1.0	57.6±1.5	58.3±1.1	**63.0±2.3**

Table 6.1: Mean accuracy of dense SIFT (DSIFT), DeCAF(FC6), VGG-VD and the Fisher convolutional autoencoder network, for parametric texture filters. With the exception of KTH and FMD datasets, the proposed Fisher convolutional features provide the best results.

Dataset	DeCAF	DSIFT + DeCAF	Fisher + DeCAF
UIUC	96.4±0.7	98.6±0.7	**98.9±0.2**
KTH-a	73.9±4.2	78.8±3.7	**81.7±2.5**
KTH-b	74.8±4.3	79.3±3.9	**84.3±1.5**
FMD	63.2±1.4	67.3±1.1	**70.7±4.6**
DTD	57.6±1.5	66.5±1.4	**68.8±3.8**

Table 6.2: Mean accuracy of the Fisher convolutional autoencoder network in concatenation with DeCAF model (pretrained on ImageNet), for parametric texture filters. This combination outperforms DeCAF and DSIFT + DeCAF features.

Dataset	VGG	DSIFT + VGG	Fisher + VGG
UIUC	96.7±1.5	98.8±0.7	**99.3±0.2**
KTH-a	73.3±3.9	77.8±4.1	**82.7±3.9**
KTH-b	73.2±3.9	78.3±2.5	**83.2±4.1**
FMD	65.5±1.0	67.5±0.9	**70.8±4.7**
DTD	58.3±1.1	67.2±1.0	**71.5±1.9**

Table 6.3: Mean accuracy of the Fisher convolutional autoencoder network in concatenation with VGG-VD model (pretrained on ImageNet), for parametric texture filters. The results are better than VGG and DSIFT + VGG, in all the datasets.

Dataset	Others	Fisher
UIUC	98.8±0.7	**99.3±0.2**
KTH-a	78.8±3.7	**82.7±3.9**
KTH-b	79.3±3.9	**84.3±1.5**
FMD	67.5±0.9	**70.8±4.7**
DTD	67.2±1.0	**71.5±1.9**

Table 6.4: The best mean accuracies of other methods (Others), in comparison with, the Fisher convolutional autoencoder network, for parametric texture filters. The proposed Fisher convolutional features outperform the best of the other approaches, with maximum margin on KTH-b dataset, and minimum improvement on the UIUC dataset.

Table 6.4 gathers the top performances from the above experiments to make an overall benchmark (Others vs. Fisher). It is worth mentioning that for KTH datasets, a combination of the proposed Fisher convolutional and DeCAF features, gives better outcomes in contrast to UIUC, FMD and DTD datasets, where VGG-VD features result in higher accuracies.

It appears that the reason for poor results on KTH dataset is due to the quality of the texture images, which were captured under controlled lighting conditions and at fixed distances (Mallikarjuna et al. [2006]). These generate more homogeneous features for each classes of textures and hence, DeCAF performs better than VGG-VD due to its pretraining specifications and deep architecture.

Tables 6.5, 6.6 and 6.7 present scales, orientations (in radiant) and resolutions (in pixel), for texture filters $\{1, 19, 37, 38, 39\}$ from LM bank, $\{1, 19, 37, 38\}$ from MR bank and $\{1\}$ from Schmid bank. These are the first filters from each texture bank that share same generation functions (Equation 5.1). These tables also show the evolution of parameters for all the illustrated filters, compared to their initial parameters. The variations in scales are considerable, whereas the resolutions correspond to the focus of the texture images.

Figure 6.4 provides some example illustrations of the initial and optimal texture filters for all the datasets, under examination. Some filters seem to remain unchanged, because their initial parameters are optimum for the texture recognition, on that specific dataset. Other alterations in, either all or part of the parameters, reflects as variations in patterns of optimized filters, compared to their initial correspondents.

Dataset	LM (1)	LM (19)	LM (37)	LM (38)	LM (39)
UIUC	2.61, 2.66, 17	1.41, 2.74, 17	3.99, 0, 49	1.41, 0, 09	4.25, 0, 09
KTH-a	1.41, 0.00, 49	1.41, 0.86, 35	1.97, 0, 43	2.00, 0, 35	5.70, 0, 35
KTH-b	1.41, 0.00, 09	1.41, 0.00, 09	1.56, 0, 09	3.21, 0, 29	4.24, 0, 49
FMD	1.41, 0.00, 49	1.99, 2.17, 23	3.99, 0, 49	1.42, 0, 49	4.24, 0, 43
DTD	1.41, 0.91, 49	2.31, 1.99, 23	3.99, 0, 49	1.41, 0, 49	4.24, 0, 49
Initials	1.41, 0.00, 49	1.41, 0.00, 49	1.41, 0, 49	1.41, 0, 49	4.24, 0, 49

Table 6.5: Examples of initial and optimal parameters (scale, orientation, and resolution) for LM texture bank. The numbers in parentheses, show the index of filters inside the bank. The greatest deviations from the initial parameters, come from filters, 37 for scale and 19 for orientation & resolution.

Dataset	MR (1)	MR (19)	MR (37)	MR (38)
UIUC	1.00, 2.09, 23	3.51, 0.00, 49	10.79, 0, 17	09.00, 0, 09
KTH-a	1.00, 0.00, 49	1.82, 0.86, 35	10.00, 0, 49	10.00, 0, 49
KTH-b	3.12, 2.22, 49	3.39, 0.00, 49	09.18, 0, 49	09.87, 0, 49
FMD	1.00, 0.48, 43	2.01, 0.54, 17	09.00, 0, 09	09.01, 0, 49
DTD	1.59, 0.87, 29	1.02, 0.76, 49	11.00, 0, 49	10.84, 0, 23
Initials	1.00, 0.00, 49	1.00, 0.00, 49	10.00, 0, 49	10.00, 0,49

Table 6.6: Examples of initial and optimal parameters (scale, orientation, and resolution) for MR texture bank. The numbers in parentheses, represent the index of filters inside the bank. The biggest variations from the initial parameters in the datasets, belong to filters, 19 for scale & resolution and 1 for orientation.

Dataset	Schmid (1)
UIUC	2.00, 1.00, 49
KTH-a	2.42, 1.18, 49
KTH-b	3.36, 1.93, 43
FMD	2.99, 2.94, 35
DTD	8.69, 3.01, 49
Initials	2.00, 1.00, 49

Table 6.7: Example of initial and optimal parameters (scale, orientation, and resolution) for Schmid texture bank. The number in parentheses, corresponds to the index of filter in the bank. The largest diversions from the initial parameters, come from scales, then orientations and finally, resolutions.

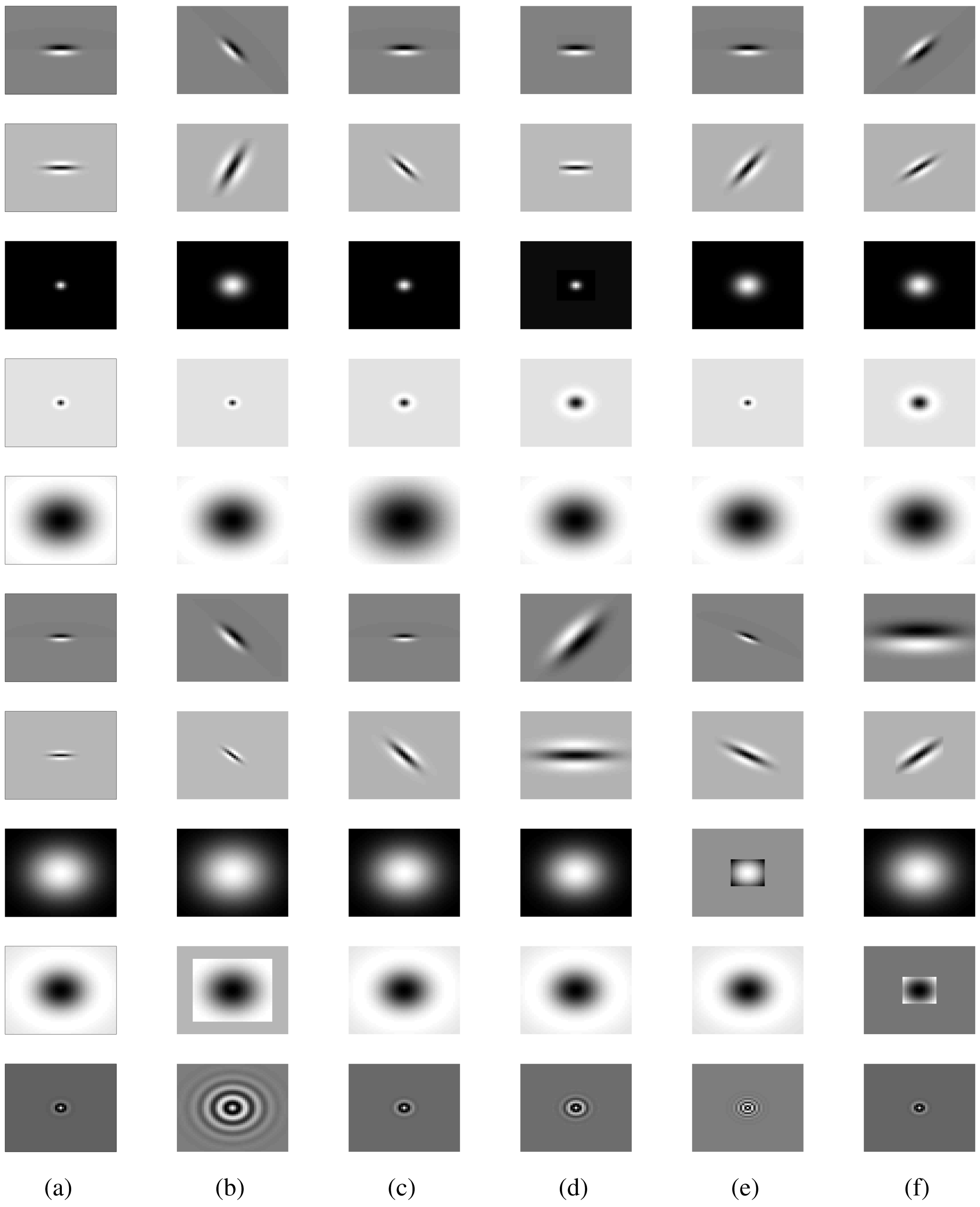

Figure 6.4: Examples of (a) initial filters and their corresponding parametric texture filters, learned for (b) UIUC; (c) KTH-TIPS2-a; (d) KTH-TIPS2-b; (e) FMD; (f) DTD datasets. The optimization mainly focuses on scale & orientation parameters for directional filters, and scale & resolution for rotational filters. The differences between initial and optimal parameters vary, with respect to, the quality of the texture images (focus & illumination) and the number of texture classes of the dataset.

6.3.2 Separable Texture Filters

Consider the set of texture filters $\mathcal{F} = \{\mathbf{F}_1, \ldots, \mathbf{F}_{|\mathcal{F}|}\}$ again but this time each filter $\mathbf{F}$, of size $r \times r$, is generated by multiplication of a vertical $\mathbf{v} \in \mathbb{R}^{r\times 1}$ and a horizontal $\mathbf{h} \in \mathbb{R}^{1\times r}$ vectors,

$$\mathbf{F} = \mathbf{v}\mathbf{h} \tag{6.12}$$

Computing of $\mathbf{v}$ and $\mathbf{h}$ form $\mathbf{F}$ is straightforward (Sironi et al. [2015]). Suppose that,

$$[\mathbf{U}, \mathbf{S}, \mathbf{V}] = svd(\mathbf{F}) \tag{6.13}$$

where $svd(.)$ stands for Singular Value Decomposition operator. Then, the vertical and horizontal vectors to separate $\mathbf{F}$, are defined as,

$$\begin{aligned} \mathbf{v} &= \sqrt{\mathbf{S}_{(1,1)}}\,\mathbf{U}_{(:,1)} \\ \mathbf{h} &= \sqrt{\mathbf{S}_{(1,1)}}\,\mathbf{V}^T_{(:,1)} \end{aligned} \tag{6.14}$$

Here, the aim is to find $\mathbf{v}$ and $\mathbf{h}$, which impose the maximum possible class separation in the output of the Fisher convolutional autoencoder network. The experiments follow the same setup as parametric texture filters and the results are reported for each experiments.

Table 6.8 represents DSIFT compared to the Fisher convolutional features. It is clear that, the proposed features outperform other descriptors, in all texture datasets except UIUC dataset. Tables 6.9 and 6.10 compare the performances of deep features (DeCAF & VGG) and their concatenations with DSIFT, and the Fisher convolutional features. It is worth mentioning that, both of the above deep descriptors are pretrained on ILSVRC dataset (Russakovsky et al. [2015]). This has a large number of samples and object classes, enabling better generalization on the larger datasets. Again, the Fisher convolutional autoencoder network outperform on all datasets except UIUC dataset, in contrast to the great improvements, with respect to, the Table 6.8. On KTH-TIPS2-a and KTH-TIPS2-b datasets, the performance of joining with DeCAF, is better than VGG. This impressive performance by the Fisher convolutional features, is due to the quality of texture images in KTH datasets, which were captured under controlled lighting conditions and fixed distances (Mallikarjuna et al. [2006]).

Dataset	DSIFT	DeCAF	VGG	Fisher
UIUC	96.6±0.7	96.4±0.7	**96.7±1.5**	89.5±0.7
KTH-a	68.4±5.6	**73.9±4.2**	73.3±3.9	70.9±4.1
KTH-b	69.0±5.7	**74.8±4.3**	73.2±3.9	69.8±0.6
FMD	50.3±1.4	63.2±1.4	**65.5±1.0**	62.2±1.9
DTD	59.6±1.0	57.6±1.5	58.3±1.1	**60.1±1.3**

Table 6.8: Mean accuracy of dense SIFT (DSIFT), DeCAF(FC6), VGG-VD and the Fisher convolutional autoencoder network, for separable texture filters. Except on DTD dataset, the proposed deep architecture is not successful to outperform others.

Dataset	DeCAF	DSIFT + DeCAF	Fisher + DeCAF
UIUC	96.4±0.7	98.6±0.7	**98.7±0.6**
KTH-a	73.9±4.2	78.8±3.7	**81.1±2.2**
KTH-b	74.8±4.3	79.3±3.9	**85.1±0.9**
FMD	63.2±1.4	67.3±1.1	**77.8±2.9**
DTD	57.6±1.5	66.5±1.4	**74.7±2.4**

Table 6.9: Mean accuracy of the Fisher convolutional autoencoder network in concatenation with DeCAF model (pretrained on ImageNet), for separable texture filters. The proposed approach greatly improves the performance on all the datasets.

Dataset	VGG	DSIFT + VGG	Fisher + VGG
UIUC	96.7±1.5	**98.8±0.7**	96.3±0.1
KTH-a	73.3±3.9	77.8±4.1	**78.4±7.4**
KTH-b	73.2±3.9	78.3±2.5	**79.1±0.6**
FMD	65.5±1.0	67.5±0.9	**72.5±2.5**
DTD	58.3±1.1	67.2±1.0	**79.6±1.3**

Table 6.10: Mean accuracy of the Fisher convolutional autoencoder network in concatenation with VGG-VD model (pretrained on ImageNet), for separable texture filters. The proposed method works better everywhere, except for UIUC dataset, containing grayscale images.

Dataset	Others	Fisher
UIUC	**98.8±0.7**	98.7±0.6
KTH-a	78.8±3.7	**81.1±2.2**
KTH-b	79.3±3.9	**85.1±0.9**
FMD	67.5±0.9	**77.8±2.9**
DTD	67.2±1.0	**79.6±1.3**

Table 6.11: The best mean accuracies of other methods (Others), in comparison with, the Fisher convolutional autoencoder network, for separable texture filters. The results confirm the superior ability of the proposed deep architecture at improving texture classification performance, on the datasets with variety of image qualities and numbers of texture classes.

In contrast, the concatenation with VGG-VD performs better than DeCAF, on FMD and DTD datasets, because here, the texture images were gathered from the web with a huge variety of lighting and capturing conditions. It seems that VGG-VD generalizes better for uncontrolled conditions in comparison with DeCAF, due to its deeper architecture.

Table 6.11 summarizes the best results from all the experiments. This confirms that the Fisher convolutional autoencoder network is quite successful at imposing distinction among highly-correlated texture patterns, especially those captured in uncontrolled conditions.

Figure 6.5 illustrates some examples of initial and their corresponding separable filters. The first column includes classic texture descriptors from aforementioned filter banks and the remaining columns, give the learned separable filters for each dataset. It can be seen that, for some filters, the changes across various datasets are smaller than for others. This means that, they are responsible for identifying common features in texture patterns. The filters with considerable deformations usually connect to deeper stacks of autoencoders, that capture higher order representations. For some of the optimized filters that are initially symmetrical, the outputs do not always end up in symmetrical filters.

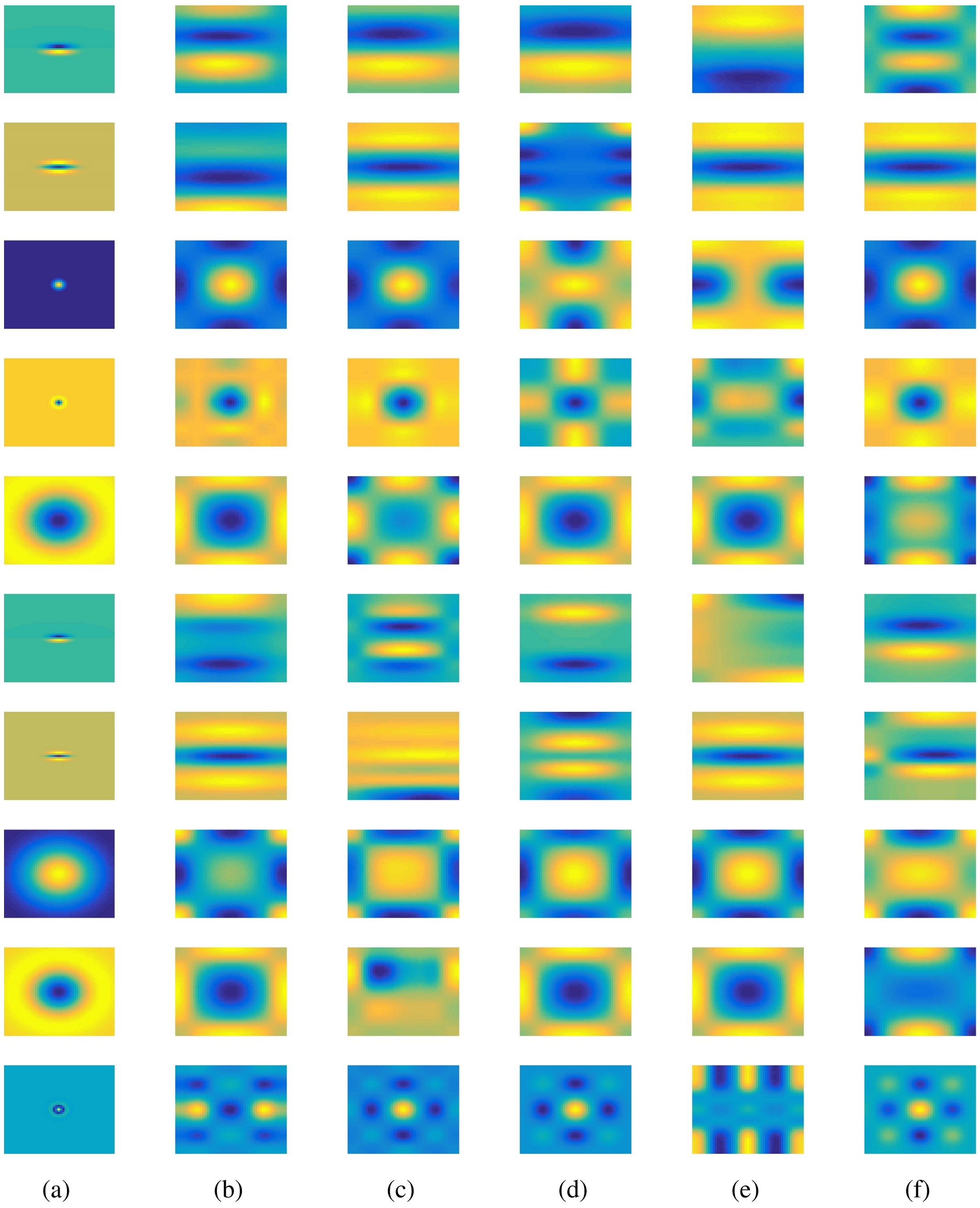

Figure 6.5: Examples of (a) initial filters and their corresponding separable texture filters, learned for (b) UIUC; (c) KTH-TIPS2-a; (d) KTH-TIPS2-b; (e) FMD; (f) DTD datasets. Since each separable filter corresponds to its optimal horizontal-vertical vectors, the variations in the learned reception fields, with respect to, the initial filters, are considerably larger than the parametric texture filters.

6.3.3 Supplementary Experiments

In order to further investigate the pipeline of Fisher convolutional autoencoder network, additional experiments are conducted to measure the contribution of each modules, *i.e.* projection-backprojection, convolution and dictionary learning to the overall performance of the texture classification task.

The first experiment removes the projection-backprojection from the proposed pipeline. Tables 6.12 show the results of this scenario. It can be seen that, they play a vital rule in improving the texture classification results for all the experimental datasets, by imposing discrimination among texture classes.

In the second experiment, the convolution operator is dropped from the Fisher convolutional autoencoder network. The results, presented in Tables 6.13, confirm that a Fisher autoencoder does better texture classification with the convolution. This is because, textures are highly-correlated patterns that need high-order convolutional features for recognition.

The third experiment removes the dictionary learning from the Fisher convolutional autoencoder network. According to the outcomes, presented in Table 6.14, it is critical for the purpose of texture classification. It seems that, when higher distinction is imposed into the convolutional features, they can generate dictionaries of higher resolutions.

6.4 Conclusion

In this chapter, coding-decoding layers of convolutional autoencoder networks were modified by the Fisher discrimination, to learn optimal parameters of texture filters and improve the classification performance. A Fisher autoencoder was composed of projection, convolution and backprojection layers. For projection and backprojection, mapping matrices were calculated by optimizing Fisher's criterion with a regularization term, targeting at better class separability. The experiments with parametric and separable texture filters showed improvements on texture classification performance. The supplementary experiments verified the contributions of all the proposed layers to the performance of the Fisher autoencoder.

Dataset	DSIFT	DeCAF	VGG	Fisher
UIUC	96.6±0.7	96.4±0.7	**96.7±1.5**	82.2±1.7
KTH-a	68.4±5.6	**73.9±4.2**	73.3±3.9	57.4±3.4
KTH-b	69.0±5.7	**74.8±4.3**	73.2±3.9	61.0±4.5
FMD	50.3±1.4	63.2±1.4	**65.5±1.0**	42.7±1.4
DTD	59.6±1.0	57.6±1.5	**58.3±1.1**	53.8±2.1

Dataset	DeCAF	DSIFT + DeCAF	Fisher + DeCAF
UIUC	96.4±0.7	**98.6±0.7**	85.2±2.3
KTH-a	73.9±4.2	**78.8±3.7**	62.6±4.6
KTH-b	74.8±4.3	**79.3±3.9**	67.3±2.8
FMD	63.2±1.4	**67.3±1.1**	58.8±6.8
DTD	57.6±1.5	**66.5±1.4**	57.1±1.2

Dataset	VGG	DSIFT + VGG	Fisher + VGG
UIUC	96.7±1.5	**98.8±0.7**	84.9±4.1
KTH-a	73.3±3.9	**77.8±4.1**	62.1±2.4
KTH-b	73.2±3.9	**78.3±2.5**	71.0±5.3
FMD	65.5±1.0	**67.5±0.9**	58.9±2.1
DTD	58.3±1.1	**67.2±1.0**	60.8±2.0

Table 6.12: Mean accuracy of the Fisher convolutional autoencoder network, without projection-backprojection. According to the results, projection-backprojection are vital modules of the proposed deep architecture, because they successfully impose greater distinction among the different texture classes. This leads to better classification, compared to the current approaches elsewhere in literature.

Dataset	DSIFT	DeCAF	VGG	Fisher
UIUC	96.6±0.7	96.4±0.7	**96.7±1.5**	71.3±3.3
KTH-a	68.4±5.6	**73.9±4.2**	73.3±3.9	51.6±5.0
KTH-b	69.0±5.7	**74.8±4.3**	73.2±3.9	55.1±6.1
FMD	50.3±1.4	63.2±1.4	**65.5±1.0**	38.7±1.6
DTD	59.6±1.0	57.6±1.5	**58.3±1.1**	46.8±2.1

Dataset	DeCAF	DSIFT + DeCAF	Fisher + DeCAF
UIUC	96.4±0.7	**98.6±0.7**	74.9±3.3
KTH-a	73.9±4.2	**78.8±3.7**	61.7±4.7
KTH-b	74.8±4.3	**79.3±3.9**	62.0±6.3
FMD	63.2±1.4	**67.3±1.1**	52.1±2.3
DTD	57.6±1.5	**66.5±1.4**	43.4±2.1

Dataset	VGG	DSIFT + VGG	Fisher + VGG
UIUC	96.7±1.5	**98.8±0.7**	75.5±4.2
KTH-a	73.3±3.9	**77.8±4.1**	58.8±2.8
KTH-b	73.2±3.9	**78.3±2.5**	64.1±4.8
FMD	65.5±1.0	**67.5±0.9**	52.2±2.1
DTD	58.3±1.1	**67.2±1.0**	47.9±1.8

Table 6.13: Mean accuracy of the Fisher convolutional autoencoder network, without convolution operator. The outcomes show that convolution of texture filters after projection and before backprojection, allows to extract higher-order features. These significantly contribute to better separability among learned dictionaries, employed for the texture classification.

Dataset	DSIFT	DeCAF	VGG	Fisher
UIUC	96.6±0.7	96.4±0.7	**96.7±1.5**	81.2±2.6
KTH-a	68.4±5.6	**73.9±4.2**	73.3±3.9	55.7±2.9
KTH-b	69.0±5.7	**74.8±4.3**	73.2±3.9	57.3±1.6
FMD	50.3±1.4	63.2±1.4	**65.5±1.0**	43.0±3.5
DTD	59.6±1.0	57.6±1.5	**58.3±1.1**	47.5±3.2

Dataset	DeCAF	DSIFT+DeCAF	Fisher + DeCAF
UIUC	96.4±0.7	**98.6±0.7**	80.8±4.1
KTH-a	73.9±4.2	**78.8±3.7**	67.4±3.5
KTH-b	74.8±4.3	**79.3±3.9**	64.4±4.8
FMD	63.2±1.4	**67.3±1.1**	56.6±5.0
DTD	57.6±1.5	**66.5±1.4**	54.9±5.2

Dataset	VGG	DSIFT+VGG	Fisher + VGG
UIUC	96.7±1.5	**98.8±0.7**	82.9±2.3
KTH-a	73.3±3.9	**77.8±4.1**	66.2±3.6
KTH-b	73.2±3.9	**78.3±2.5**	67.8±2.5
FMD	65.5±1.0	**67.5±0.9**	56.7±5.2
DTD	58.3±1.1	**67.2±1.0**	55.6±4.8

Table 6.14: Mean accuracy of the Fisher convolutional autoencoder network, without dictionary learning. Based on the experimental results, dictionary learning is the best classification approach to be taken for highly-correlated textures. This supremacy comes from better discrepancy in the convolutional features, imposed by the projection-backprojection, that eventually, generates high-distinction dictionaries.

Part III

Object Recognition

Chapter 7

Multipartite Pooling

The considerable complexity of object recognition makes it an interesting research topic in computer vision. Deep neural networks have recently addressed this challenge, with close precision to human observers. They recognize thousands of objects from millions of images, by using the models with large learning capacity. This chapter proposes a novel pooling strategy that learns how to rank convolutional features adaptively, allowing the selection of more informative representations.

To this end, the Fisher discrimination for dimension expansion is exploited, to project the features into a space, spanned by the number of classes in the dataset under study. This mapping is employed as a measure to rank the existing features, with respect to their specific discriminant power, for each classes. Then, multipartite ranking is applied to score the separability of instances, and to aggregate one-versus-all scores, giving an overall distinction score for each features. For the pooling, features with the highest scores are picked in a pooling window, instead of maximum, average or stochastic random assignments.

Spatial pooling of convolutional features, is critical in many deep neural networks. Pooling aims to select and aggregate features over a local reception field, into a local bag of representations, that are compact and resilient to transformations and distortions of the input (Boureau et al. [2010]). Common pooling strategies often take sum (Fukushima [1988]), average (Le Cun et al. [1990]), or maximum (Jarrett et al. [2009]) response. There are also variants that enhance maximum pooling performance, such as, generalized maximum pooling (Murray and Perronnin [2014]) or fractal maximum pooling (Graham [2014]). Deterministic pooling can be extended to stochastic alternatives, e.g. random selection of an activation, according to a multinomial distribution (Zeiler and Fergus [2013]).

7.1 Method

There exists a vast literature on instance selection and feature ranking. Instance selection regimes usually belong to either condensation or edition proposals (Leyva et al. [2015]). They attempt to find a subset of data, in which, a trained classifier is provided with, similar or close validation error, as the primary data. Condensed Nearest Neighbour (CoNN) (Hilborn [1968]), searches for a consistent subset, where every instance inside is assumed to be correctly classified. Some variants of this method are Reduced Nearest Neighbour (RNN) (Gowda and Krishna [1979]), Selective Nearest Neighbour (SNN) (Ritter et al. [1975]), Minimal Consistent Set (MCS) (Dasarathy [1994]), Fast Nearest Neighbour Condensation (FNNC) (Angiulli [2007]), and Prototype Selection by Clustering (PSC) (Olvera-López et al. [2010]).

In contrast, Edited Nearest Neighbour (ENN) (Wilson [1972]), discards the instances that disagree with the classification responses of their neighbouring instances. Some revisions of this strategy, are Repeated Edited Nearest Neighbour (RENN) (Tomek [1976]), Nearest Centroid Neighbour Edition (NCNEdit) (Sánchez et al. [2003]), Edited Normalized Radial Basis Function (ENRBF) (Jankowski and Grochowski [2004]), and Edited Nearest Neighbour Estimating Class Probabilistic and Threshold (ENNth) (Vázquez et al. [2005]).

On the other hand, the family of feature ranking algorithms can be mainly grouped into, preference learning, bipartite, multipartite, or multilabel ranking. In situations where the instances have only binary labels, the ranking is called bipartite. Different aspects of bipartite ranking have been investigated in numerous studies including, RankBoost (Freund et al. [2003]), RankNet (Burges et al. [2005]), and AUC maximizing SVM (Brefeld and Scheffer [2005]), which are the ranking versions for AdaBoost, logistic regression, and SVM.

There are also several ranking measures, such as, average precision and Normalized Discounted Cumulative Gain (NDCG). For multilabel instances, multipartite ranking seeks to maximize the volume under the ROC surface (Waegeman and De Baets [2011]), which is in contrast with the minimization of the pairwise ranking cost (Uematsu and Lee [2015]).

The problem of employing either instance selection or feature ranking methods for pooling in deep neural networks, appears at the testing phase of trained models. The existing ranking algorithms, mostly deals with the training-time ranking. As a result, they are not usually advantageous for the pooling of convolutional features in the test phase. Without pooling, the performance of deep learning architectures degrades substantially. The local feature responses propagate less effectively to neighbouring receptive fields, thus the local-global representation power of the convolutional network diminishes. Moreover, the network becomes very sensitive to input deformations.

To tackle the above issues, a novel strategy *i.e.* multipartite pooling, is introduced. This ranks convolutional features by employing supervised learning. In supervised learning, the trained scoring function reflects the ordinal relation among class labels. The multipartite pooling scheme learns a projection from the training set. Intuitively, this is a feature selection operator, whose aim is to pick the most informative convolutional features, by learning a multipartite ranking scheme from the training set. Inspired by stochastic pooling, higher ranked activations in each window, are picked with respect to their scoring function responses. Since this multipartite ranking is based on the class information, it can generate a coherent ranking of features, for both of the training and test sets. This also leads to an efficient spread of responses, and effective generalization for deep convolutional neural networks.

In summary, the proposed multipartite pooling method has several advantages. This considers the distribution of each class and calculates the rank of individual features. Due to the data-driven process of scoring, the performance gap between training-test errors, is considerably closer. It also generates superior performance on standard benchmark datasets, in comparison with the average, maximum and stochastic pooling schemes, when identical evaluation protocols are applied. The conducted experiments on various benchmarks, confirm that the proposed strategy of multipartite pooling, consistently improves the performance of deep convolutional networks, by using better model generalization for the test-time data.

7.2 Formulation

This section begins with multipartite ranking and moves towards the multipartite pooling. The multipartite ranking means scoring of each representation in the feature set, with respect to the distinctive information. Instances with higher scores are expected to be more informative, and hence, receive higher ranks. The intuition of multipartite pooling is about picking the activation instances with the higher scores (ranks) in a pooling window, to achieve better activations in the pooling layer. A graphical interpretation of feature vs instance ranking is depicted in Figure 7.1, where columns represent the activations.

For a two-class regime, the criterion to calculate the significance of each feature, can be selected from statistical measures, such as, absolute value two-sample t-test with pooled variance estimate (Jain et al. [2003]); relative entropy or Kullback-Leibler distance (Hershey and Olsen [2007]); minimum attainable classification error or Chernoff bound (Nussbaum and Szkoła [2009]); area between the empirical Receiver Operating Characteristic (ROC) curve and the random classifier slope (Fawcett [2006]); and absolute value of the standardized u-statistic of a two-sample unpaired Wilcoxon test or Mann-Whitney test (Bohn and Wolfe [1994]).

Suppose that a set of instances $\mathcal{X} = \{X_1, \ldots, X_{|\mathcal{X}|}\}$ are assigned to a predefined label set $\mathcal{L} = \{L_1, \ldots, L_{|\mathcal{L}|}\}$, such that, $\mathcal{X}$ is a matrix with $|\mathcal{X}|$ instances (rows). The aim is to rank the features (columns), using an independent evaluation criterion. This criterion is a distance that measures the significance of an instance, for imposing higher class distinction in the set $\mathcal{X}$. The absolute value of the criterion for a bipartite ranking scenario, with only two valid labels $\{L_1, L_2\}$, is defined as,

$$\mathcal{C}_B(\mathcal{X}) = KL_{12}(X_1, \ldots, X_{|\mathcal{X}|}) \tag{7.1}$$

where KL is the Kullback-Leibler divergence and $\mathcal{C}_B(\mathcal{X})$ is the binary criterion measured for each feature (column) of the set $\mathcal{X}$. This equation can be extended to the summation of binary criteria, where each labels is considered as primary label (foreground) and the rest are merged as secondary labels (background).

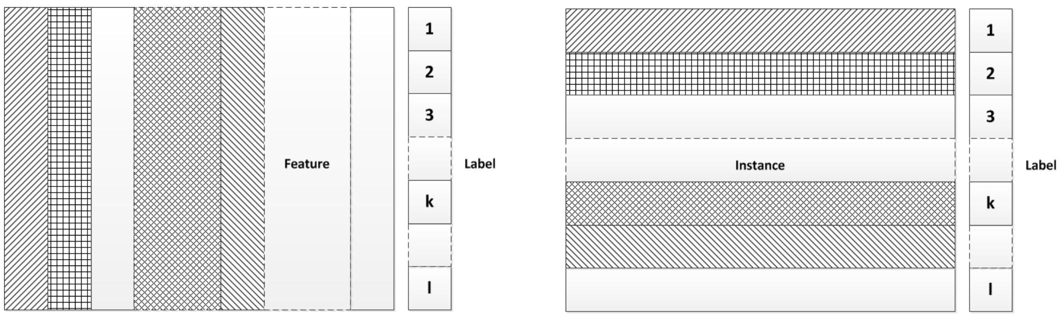

Figure 7.1: Feature vs instance ranking. A set of features (columns) and instances (rows) are assigned to $|\mathcal{L}|$ different labels. They are ranked upon their separability, represented by different line patterns and are scored. These scores are used for selecting the best features or instances. To employ either of them for convolutional pooling, the labels must be known. The problem is that, classic feature-instance ranking methods are specific to the training-time data, and there is no way to exploit them for pooling of the test-time data. To solve this inconsistency, the notion of labels is mapped to the test data and then, instance ranking strategies are applied to the pooling layers. This is accomplished by the supervised projection.

The overall criterion of the multipartite case, with multiple labels $\mathcal{L}$, can be formulated as,

$$\mathcal{C}_M(\mathcal{X}) = \sum_{i=1,\, j\neq i}^{i=|\mathcal{L}|} KL_{ij}(X_1, \ldots, X_{|\mathcal{X}|}) \tag{7.2}$$

where KL_{ij} is the cumulative Kullback-Leibler distance of label L_i to the rest of the labels of set $\mathcal{L}$, which are $\forall\ L_j \in \mathcal{L} - L_i$. It is clear that, higher values of $\mathcal{C}_M(\mathcal{X})$ for a feature, means better class separability. A high-ranked representation is beneficial to any classifier, because there are better distinctions between classes in the set $\mathcal{X}$.

It is possible to employ the above formulation for instance ranking. In other words, instances (rows) are ranked instead of features (columns), which is required for pooling operation, where high-ranked instances are selected as the representations for each convolutional filters. In contrast with feature ranking, the rows of set $\mathcal{X}$, which correspond to convolutional representations, are ranked in the pooling layers.

To connect the features into instances, a projection from the feature space into a new instance space, spanned by the number of classes in $\mathcal{X}$, is employed. In this space, a new set $\mathcal{P}$ is created by multiplying the feature set $\mathcal{X}$ with a projection matrix $\mathbf{A}$, such that,

$$\mathcal{C}_M(\mathcal{P}) = \sum_{i=1,\, j\neq i}^{i=|\mathcal{L}|} KL_{ij}(X_1\mathbf{A}, \ldots, X_{|\mathcal{X}|}\mathbf{A}) \tag{7.3}$$

where the set $\mathcal{P}$ is a matrix with $|\mathcal{X}|$ instance (rows) and $|\mathcal{L}|$ features (columns). The projection matrix $\mathbf{A}$ enables the same ranking strategies for features of the set $\mathcal{X}$, to be applied to the instances of the $\mathcal{P}$, so that, the highly-ranked activations are selected.

7.2.1 Supervised Projection

To formulate the above projection, the matrix $\mathbf{A}$ can be considered as a mapping, which tries to project $\mathcal{X}$ into a space with $c = |\mathcal{L}|$ dimensions. The projection matrix $\mathbf{A}$ is determined to minimize the Fisher criterion given by,

$$\mathcal{J}(\mathbf{A}) = tr\left[(\mathbf{A}\mathbf{S_w}\mathbf{A}^T)(\mathbf{A}\mathbf{S_b}\mathbf{A}^T)^{-1}\right] \tag{7.4}$$

that $tr(.)$ is the diagonal summation operator. The within ($\mathbf{S_w}$) and between ($\mathbf{S_b}$) class scatterings are defined as,

$$\mathbf{S_w} = \sum_{j=1}^{c} \sum_{x_i \in \mathbf{C}_j} (x_i - \mu_j)(x_i - \mu_j)^T \tag{7.5}$$

$$\mathbf{S_b} = \sum_{j=1}^{c} (\mu_j - \bar{\mu})(\mu_j - \bar{\mu})^T \tag{7.6}$$

where c, μ_j and $\bar{\mu}$ are number of classes, mean over class $\mathbf{C}_j$ and mean over the set $\mathcal{X}$. The matrix $\mathbf{S_w}$ can be regarded as average class-specific covariance, whereas $\mathbf{S_b}$ can be viewed as the mean distance between all different classes. The purpose of Equation 7.4 is to maximize the between-class scattering, while preserving within-class dispersion. The solution can be retrieved from a generalized eigenvalue problem $\mathbf{S_b}\mathbf{A} = \lambda\mathbf{S_w}\mathbf{A}$. For $c = |\mathcal{L}|$ classes, the projection matrix $\mathbf{A}$ builds upon eigenvectors corresponding to the largest c eigenvalues of $\mathbf{S_w^{-1}}\mathbf{S_b}$ (Bishop [2006]).

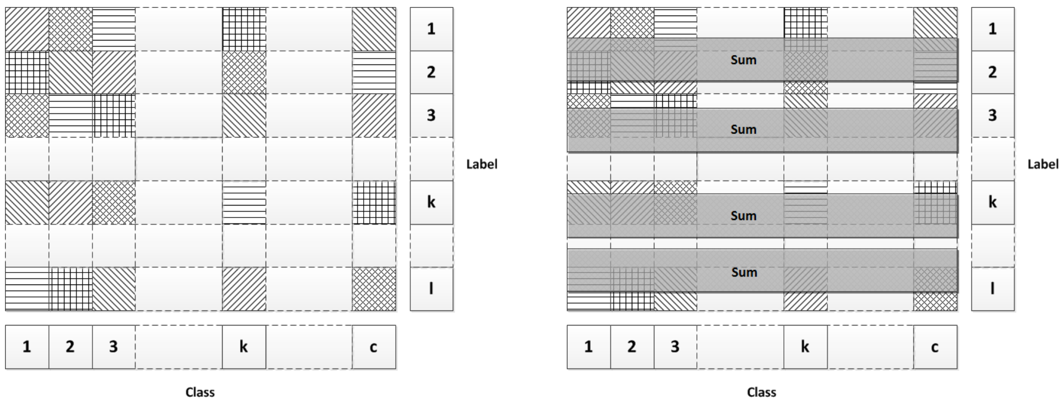

Figure 7.2: Multipartite ranking. The projected set $\mathcal{P}$ is ranked and the scores are aggregated to compute overall criterion $\mathcal{C}_M(\mathcal{P})$. Since the columns represent classes inside the feature set $\mathcal{X}$, bipartite rank of each columns, is calculated with respect to the rest of columns. This generates $c = |\mathcal{L}|$ different scores, represented by different line patterns, for each of the $|\mathcal{P}|$ instances. By sliding an accumulative bar, represented by grey rectangle, the overall score is computed for each instances. These overall scores are used to rank and pool the most informative instances, which are activations of the pooling layers.

To yield better distinction, the ratio of between and within class scatterings (quotient-of-trace) is minimized (Cunningham and Ghahramani [2015]), by imposing orthogonality to the following cost function,

$$\mathcal{Q}(\mathbf{A}) = tr(\mathbf{A}\mathbf{S_w}\mathbf{A^T})\left[tr\big(\mathbf{A}\mathbf{S_b}\mathbf{A^T}\big)\right]^{-1} + \lambda\|\mathbf{I} - \mathbf{A}\mathbf{A^T}\|_2 \qquad (7.7)$$

The first part of function $\mathcal{Q}(\mathbf{A})$ defines a form of Fisher criterion that aims for making the highest possible separability among classes. The second term is a regularization term imposing orthogonality into the projection matrix $\mathbf{A}$. Looking back at Equation 7.4, it can be seen that the set of eigenvectors corresponding to the largest c eigenvalues of $\mathbf{S_w^{-1}S_b}$ is a solution for the above optimization problem. This can be taken as an initial projection matrix $\mathbf{A^{(0)}}$.

Now, it is possible to minimize $\mathcal{Q}(\mathbf{A})$ by using stochastic gradient descent. Here, $\mathbf{A^{(0)}}$ is employed as an initialization point, because conventional Fisher criterion is the trace-of-quotient, which can be solved by generalized eigenvalue method. Equation 7.7 is the quotient-of-trace, that requires a different solution (Cunningham and Ghahramani [2015]).

Algorithm 4 Supervised Projection

```
Input: feature set $\mathcal{X}$
Output: projection matrix $\mathbf{A}$

1: Compute $\mathbf{S_w}$ (Equation 7.5) and $\mathbf{S_b}$ (Equation 7.6)
2: Set $\mathbf{A}^{(0)}$ as largest eigenvalues of $\mathbf{S_w^{-1}S_b}$
3: Minimize Equation 7.7 by using Equation 7.10
```

To work out the closed form derivatives of Equation 7.7, suppose that $\mathcal{Q}(\mathbf{A})$ is composed of $\mathcal{Q}_1(\mathbf{A})$ and $\mathcal{Q}_2(\mathbf{A})$ as follows,

$$\mathcal{Q}_1(\mathbf{A}) = tr(\mathbf{A}\mathbf{S_w}\mathbf{A^T})\left[tr(\mathbf{A}\mathbf{S_b}\mathbf{A^T})\right]^{-1} \tag{7.8}$$

$$\mathcal{Q}_2(\mathbf{A}) = \lambda\|\mathbf{I} - \mathbf{A}\mathbf{A^T}\|_2 \tag{7.9}$$

According to matrix calculus (Petersen et al. [2008]),

$$\begin{aligned} \frac{\partial \mathcal{Q}}{\partial \mathbf{A}} &= \frac{tr(\mathbf{A}\mathbf{S_w}\mathbf{A^T})}{\left[tr(\mathbf{A}\mathbf{S_b}\mathbf{A^T})\right]^2}(\mathbf{S_b^T} + \mathbf{S_b})\mathbf{A^T} \\ &- \frac{1}{tr(\mathbf{A}\mathbf{S_b}\mathbf{A^T})}(\mathbf{S_w^T} + \mathbf{S_w})\mathbf{A^T} \\ &- \frac{2\lambda}{\|\mathbf{I} - \mathbf{A}\mathbf{A^T}\|_2}\mathbf{A^T}(\mathbf{I} - \mathbf{A}\mathbf{A^T}) \end{aligned} \tag{7.10}$$

The computation of $\mathbf{A}$ is summarized in Algorithm 4. For implementation purposes, the built-in function of Matlab optimization toolbox (Coleman and Li [1996]) is employed.

7.2.2 Multipartite Ranking

Drawing upon the above information, it is possible to put forward the proposed multipartite ranking scheme. Using the instance ranking strategy, one can take the feature set $\mathcal{X}$, deploy the supervised projection $\mathbf{A}$ (Algorithm 4) to produce the projected set $\mathcal{P}$, and calculate the cumulative Kullback-Leibler distance (Equation 7.3), as the ranking scores, for each instance of the projected set $\mathcal{P}$.

Algorithm 5 Multipartite Ranking

```
Input: feature set $\mathcal{X}$, label set $\mathcal{L}$
Output: overall criterion $\mathcal{C}_M(\mathcal{P})$

1: Compute the projection matrix $\mathbf{A}$ (Algorithm 4)
2: Calculate the projected set $\mathcal{P} = \mathcal{X}\mathbf{A}$

for $i = 1$ to $|\mathcal{L}|$ do
    3: Split $\mathcal{P}$ between labels $\{L_i, L_j\}$, when $L_j \in \mathcal{L} - L_i$
    4: Calculate $KL_{ij}(\mathcal{P})$ (Equation 7.3)
end for

5: Set $\mathcal{C}_M(P) = \sum KL_{ij}(\mathcal{P})$
```

Since the number of instances in $\mathcal{P}$ is equal to the number of instances in $\mathcal{X}$, and these two matrices are related linearly through $\mathbf{A}$ via $\mathcal{P} = \mathcal{X}\mathbf{A}$, the overall criterion $\mathcal{C}_M(\mathcal{P})$ is sorted to rank the instances of the set $\mathcal{X}$, in regard to their class separability. Algorithm 5 represents the multipartite ranking method. The process of the multipartite ranking is also visualized in Figure 7.2. Each column of the set $\mathcal{P}$ represents a specific class of the set $\mathcal{X}$ and hence, the Kullback-Leibler binary scoring scheme (one-versus-all) is employed to set a criterion measure for each of its individual instances (rows). After it has been applied to all the columns, it starts to scan rows and accumulate scores, resulting in the overall criteria, $\mathcal{C}_M(\mathcal{P})$. This is then used to rank the instances of the projected set $\mathcal{P}$.

The reason for projecting to the space spanned by the number of classes, is to use the above, one-versus-all strategy. The bipartite ranking by Kullback-Leibler divergence requires that, one main class is selected as the foreground label, while those remaining are used as background labels. It gives a measure of how the foreground is separated from the background data. It is necessary to use statistics to ensure that the cumulative criterion, $\mathcal{C}_M(\mathcal{P})$ is a true representation of the all instances, contained within the set $\mathcal{X}$.

When $\mathcal{X}$ is projected to lower dimensions than the number of available classes, the result is that, some of the classes are missed. In contrast, projection of $\mathcal{X}$ to higher dimensions than the number of classes, leads to partitioning of some classes to pseudo labels, that are not queried during the test phase. Either way, the generated scores are not reliable for the sake of pooling, because they are not derived from the actual distribution of the classes.

Algorithm 6 Multipartite Pooling

```
**Input:** convolutional feature stack $\mathcal{S} \in \mathbb{R}^{h\times w\times d\times n}$
**Output:** overall criterion $\mathcal{C}_M(\mathcal{S})$

1: Reshape each $S_i \in \mathbb{R}^{h\times w\times d}$ to $S_i \in \mathbb{R}^{hw\times d}$
2: Concatenate all $S_i$ columns to give $\mathcal{X} \in \mathbb{R}^{hwn\times d}$
3: Calculate $\mathcal{C}_M(\mathcal{P}) \in \mathbb{R}^{hwn\times 1}$ by Algorithm 5
4: Partition $\mathcal{C}_M(\mathcal{P})$ to give $\mathcal{C}_M(\mathcal{S}) \in \mathbb{R}^{h\times w\times n}$
5: Pool the activations based on $\mathcal{C}_{M_i} \in \mathbb{R}^{h\times w}$ for all $i \in [1:n]$
```

7.2.3 Multipartite Pooling

The above multipartite ranking strategy can be employed for the pooling. In general, a deep convolutional neural network consists of consecutive convolution and pooling layers. The convolutional layers extract common patterns within local patches. Then, a nonlinear elementwise operator is applied to the convolutional features, and the resulting activations are passed to the pooling layers. These activations are less sensitive to the precise locations of structures within the data, than the primary features. Therefore, the consecutive convolutional layers can extract features, that are not susceptible to spatial transformations or distortions of the input (Zeiler and Fergus [2013]).

Suppose that a stack of convolutional features $\mathcal{S} = \{S_1, \ldots, S_{|\mathcal{S}|}\}$ passes through the pooling layer. The matrix $\mathcal{S}$ is an array of dimensions $h \times w \times d \times n$ where h and w are height and width of samples, d is depth of stack (number of filters), such that $S_i \in \mathbb{R}^{h\times w\times d}$ creates a three-dimensional vector, and n is number of samples in the stack ($n = |\mathcal{S}|$).

The standard pooling methods either retain the maximum or average value, over the pooling region per channel. The multipartite pooling method begins with the reshaping of feature stack $\mathcal{S}$ to form n two-dimensional $X_i \in \mathbb{R}^{hw\times d}$. The elements of set $\mathcal{X} = \{X_1, \ldots, X_{|\mathcal{X}|}\}$ are concatenated such that $|\mathcal{X}| = n$ and $\mathcal{X} \in \mathbb{R}^{hwn\times d}$ is a two-dimensional matrix. Now, $\mathcal{X}$ is ready to deploy Algorithm 5 and compute the overall criterion $\mathcal{C}_M(\mathcal{P}) \in \mathbb{R}^{hwn\times 1}$. Partitioning to n and reshaping into $h \times w$ windows, give $\mathcal{C}_M(\mathcal{S}) = \{\mathcal{C}_{M_1}, \ldots, \mathcal{C}_{M_{|\mathcal{S}|}}\}$, that $\mathcal{C}_{M_i} \in \mathbb{R}^{h\times w}$ provides the rank of each pixels of X_i. For pooling, a sliding window goes through each region and picks the representation with the greatest $\mathcal{C}_M(\mathcal{S})$. These are the activations with the best separation among available classes.

As a numerical example, consider MNIST dataset with $|\mathcal{L}| = 10$ classes. The first pooling layer is fed by a stack $\mathcal{S}$, consisting of the convolutions of $d = 20$ filters, with $n = 100$ frames, of size $w = 24$ by $h = 24$ pixels. First, $\mathcal{S}$ is reshaped to 100 samples of size 576×20 pixels, form the set $\mathcal{X}$, which will be concatenated as a 57600×20 array. Second, the projection matrix $\mathbf{A} \in \mathbb{R}^{20\times10}$ is calculated to project $\mathcal{X}$. Third, $\mathcal{C}_M(\mathcal{P}) \in \mathbb{R}^{57600\times1}$ is computed and partitioned into 100 criterion measures $\mathcal{C}_{M_i}$, of size 24×24 pixels. For the pooling of $\mathcal{S}$, a 2×2 window moves along each frame and picks the top-score pixels. The output is a set of 100 features of size 12×12 pixels for each of 20 convolutional filters.

7.3 Experiments

For evaluation purposes, the multipartite pooling is compared with the popular maximum, average and stochastic poolings. A standard experimental setup (Zeiler and Fergus [2013]) is followed to apply the multipartite pooling for MNIST, CIFAR and Street View House Numbers (SVHN) datasets. The results show that, when multipartite pooling is employed to pool convolutional features, lower test error rates than other pooling strategies, are achieved. For implementation, the library provided by the Oxford Visual Geometry Group (Vedaldi and Fulkerson [2008]) is used.

7.3.1 Datasets

The MNIST dataset (LeCun et al. [1998]), contains $60,000$ training examples, and $10,000$ test samples, normalized to 20×20 pixels, centred by centre of mass in 28×28 pixels, and sheared by horizontally shifting, such that, the principal axis is vertical. The foreground pixels were set to one, and the background to zero.

The CIFAR dataset (Krizhevsky and Hinton [2009]), includes two subsets. The first subset, CIFAR-10 consists of 10 classes of objects with $6,000$ images per class. The classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. It was divided into $5,000$ randomly selected images per class as training set, and the rest served as test samples. The second subset, CIFAR-100 has 600 images for each of 100 classes. These classes also come in 20 super-classes, each consisting of five classes.

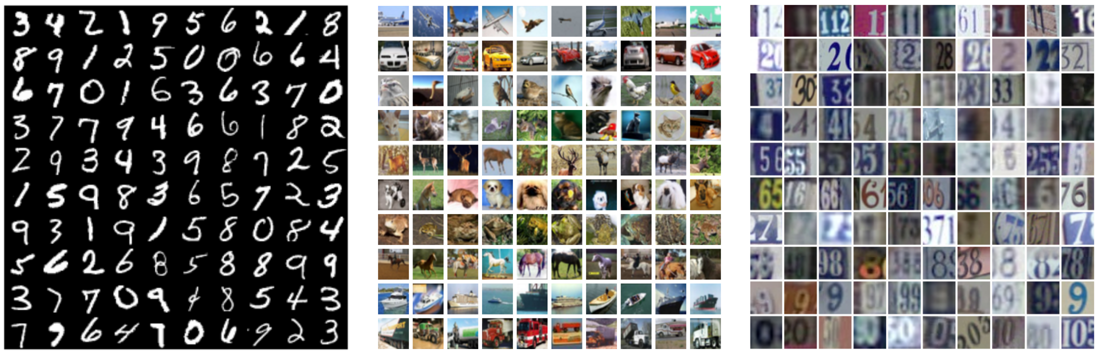

Figure 7.3: Examples of images from MNIST, CIFAR and SVHN Datasets.

The SVHN dataset (Netzer et al. [2011]), was extracted from a large number of Google Street View images by automated algorithms and the Amazon Mechanical Turk (AMT). It consists of over $600,000$ labelled characters in full numbers and MNIST-like cropped digits in 32×32 pixels. Three subsets are available containing $73,257$ digits for training, $26,032$ for testing and $531,131$ extra samples.

7.3.2 Results & Discussion

The following tables represent the evaluation results for four different schemes, including max, average, stochastic, and multipartite poolings, in terms of the training and test errors. To gain an insight into added computational load, one should recall that the only intensive calculations are related to working out the supervised projection, at the training phase. This procedure (Algorithm 4), depends on the number of activations ($h \times w \times n$ instances) and convolutional filters (d dimensions), which are employed in Algorithm 6. For testing, one just multiplies and no computational cost is involved. For example, the number of samples, trained in an identical architecture for MNIST dataset, are 285, and 195 samples per second for the max and multipartite pooling strategies, respectively.

The classification performance on MNIST dataset is reported in Table 7.1. It can be seen that, max pooling gives the best training performance, but its test error is larger than the stochastic and multipartite pooling. In other words, it may overfit for MNIST, although its test performance is higher than the average pooling. The multipartite pooling performs better than all other schemes, despite greater training error, compared to max and stochastic pooling.

Pooling	Train (%)	Test (%)
Average	0.57	0.83
Max	**0.04**	0.55
Stochastic	0.33	0.47
Multipartite (Proposed)	0.38	**0.41**

Table 7.1: Classification errors for different pooling strategies on MNIST dataset. The multipartite pooling approach provides the lowest test error, in spite of high training error. This is due to better generalization of the proposed pooling, compared to the other methods.

Pooling	Train (%)	Test (%)
Average	1.92	19.24
Max	**0.0**	19.40
Stochastic	3.40	15.13
Multipartite (Proposed)	12.63	**13.45**

Pooling	Train (%)	Test (%)
Average	11.20	47.77
Max	**0.17**	50.90
Stochastic	21.22	42.51
Multipartite (Proposed)	36.32	**40.81**

Table 7.2: Classification errors for different pooling strategies on CIFAR-10 and CIFAR-100 datasets. The multipartite pooling outperforms other pooling schemes on test errors, but it is behind on training errors. The close gap between training and test errors, leads to better classification performance for the proposed pooling strategy.

Pooling	Train (%)	Test (%)
Average	1.65	3.72
Max	**0.13**	3.81
Stochastic	1.41	2.80
Multipartite (Proposed)	2.09	**2.15**

Table 7.3: Classification errors for different pooling strategies on SVHN dataset. The multipartite pooling scheme gives the best performance on the test, that is closely followed by the training error.

The multipartite pooling is more competent because it draws upon the statistics of training-test data for pooling. It is in contrast to picking the maximum response (max pooling); smoothing the activations (average pooling); and random selection (stochastic pooling); which disregard the data distribution. In a pooling layer of any deep learning architecture, aggregation of the best available features is critical to infer complicated contexts, such as, shape derived from primitives of edge and corners. The proposed pooling learns how to pick the best informative activations from the training set, and then, employs this knowledge at the test phase. As a result, the performance consistently improves in all the experiments.

Figures 7.4 and 7.5 show the training-test performances of MNIST dataset for 20 epochs. Except for the early epochs, they are quite close to each other in the multipartite pooling regime. The reason is that, both of the training-test poolings are connected with a common factor; the projection matrix **A**. This is trained with the training set and is deployed by multipartite pooling on the test set, to pick the most informative activations. Since the same criterion (Kullback-Leibler) is employed to rank the projected activations for training-test, and they are mapped with the same projection matrix **A**, it is expected that the trained network will demonstrate better generalization than alternative pooling schemes, where the training-test poolings are statistically disconnected. The graphs show that the multipartite pooling generalizes considerably better.

Tables 7.2 provides the performance metrics for CIFAR datasets. It is apparent that the multipartite pooling outperforms other approaches on the test performance. It also prevents the model from overfitting, in contrast to the max pooling. In the proposed pooling method, better generalization also contributes to another advantage; preventing under-overfitting. As mentioned before, pooling at the test phase is linked to the training phase by the projection. This ensures that the test performance follows the training closely and hence, it is less likely to end up under-overfitting.

One striking observation is that the gap between the training-test errors is wider for CIFAR-100 compared to CIFAR-10. This relates to the number of classes in each subsets of CIFAR. Since CIFAR-100 has more classes, it is more difficult to impose separability, hence, the difference between the training and test performances will increase. Figures 7.6 and 7.7 depicts the errors of CIFAR-10 dataset for 50 epochs. It is clear that employing of the max pooling results in a huge gap between the training-test performances due to the overfitting.

Finally, the evaluation outcomes for SVHN dataset are presented in Table 7.3. Again, the multipartite pooling does a better job on the test. The error is close to the stochastic pooling method. This implies that when the number of samples in a dataset increases greatly, the multipartite ranking scores lean towards the probability distribution, generated by the stochastic pooling. Here, any infinitesimal numerical errors may also lead to an inaccurate pooling, which may, in return, degrade the informative activations of a layer for both of the pooling methods. Since the multipartite pooling takes a deterministic approach, the effect of numerical inconsistencies, is considerably smaller than stochastic pooling, which randomly picks activations on a multinomial distribution (Zeiler and Fergus [2013]).

Overall, the employment of multipartite ranking for the purpose of pooling in deep convolutional neural networks, produces superior results compared to all the other strategies, tested in the experiments. It is robust to overfitting and shows better generalization characteristics, by connecting the training and test statistics.

7.4 Conclusion

In this chapter, a new pooling scheme for deep convolutional neural networks was proposed, using Fischer discrimination to select the optimal features which highly contributed to the classification performance. Instead of using a regular pooling scheme, this algorithm aimed at actively searching of the most discriminative features in every pooling layer. The multipartite pooling only required the training data to come up with the pooling layers and hence, its extra computational complexity was restricted to the training phase. The experiments conducted on three well-known dataset (MNIST, CIFAR and SVHN) for image classification, demonstrated improved performance over the widely-used maximum, minimum, and stochastic pooling regimes. The results also showed that the proposed pooling scheme was beneficial to prevent overfitting, while imposed a better regularization and higher generalization, without the fine-tuning of any hyper-parameters.

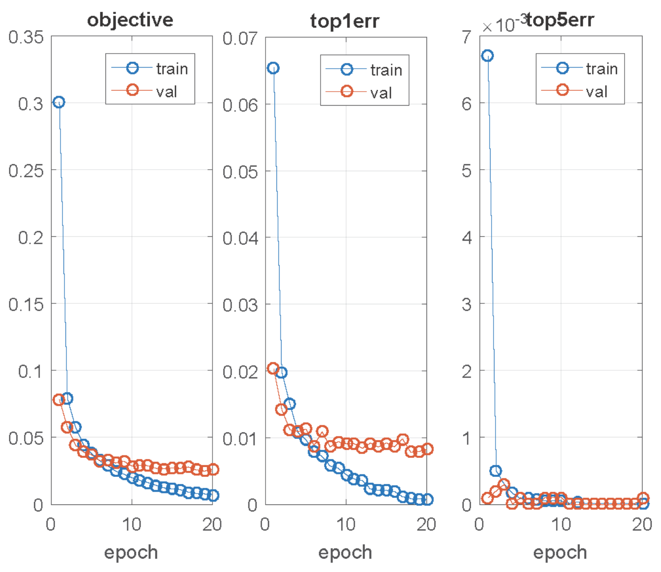

Figure 7.4: Max pooling for MNIST. The first graph represents the loss function (objective) for both training (train) and test (val) on MNIST dataset. The other graphs correspond to the top 1 (top1err) and the top 5 (top5err) errors.

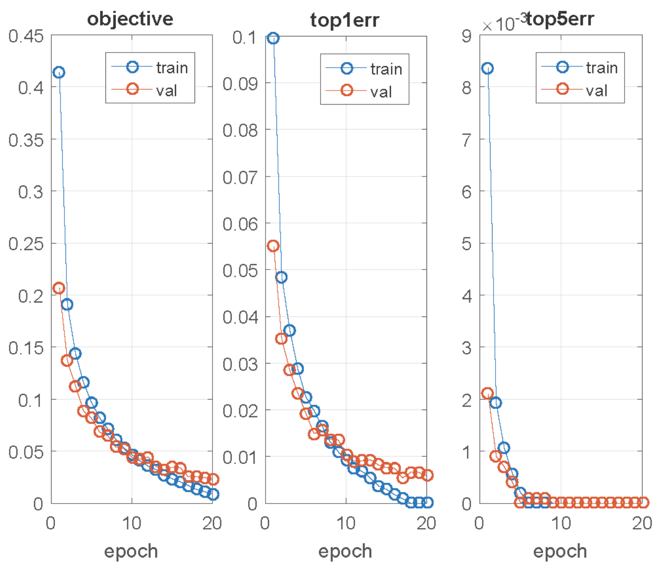

Figure 7.5: Multipartite pooling for MNIST. In comparison with the max pooling (Figure 7.4), the test loss and errors (val) are reduced by applying the multipartite pooling technique. Note that the training and test performances get closer to each other, indicating better generalization of the trained network.

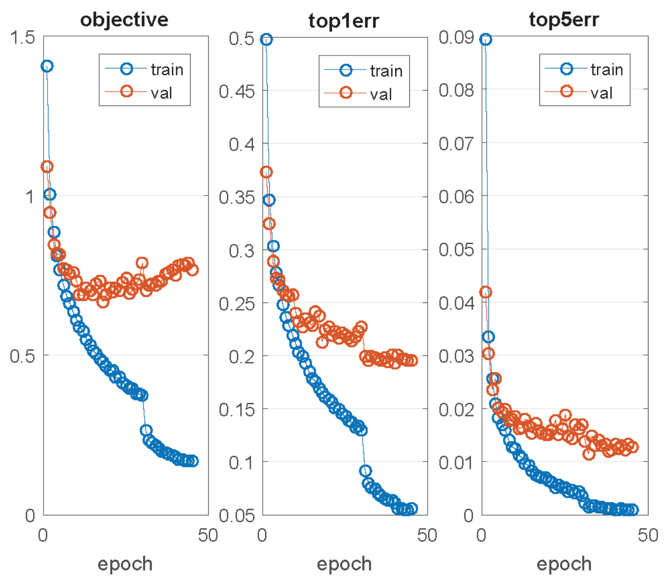

Figure 7.6: Max pooling for CIFAR-10. Compared to MNIST, this results in greater loss and errors on CIFAR-10, due to the variety in samples and tasks (character vs object recognition). The gap between the training (train) and test (val) errors, is considerably wider for CIFAR-10.

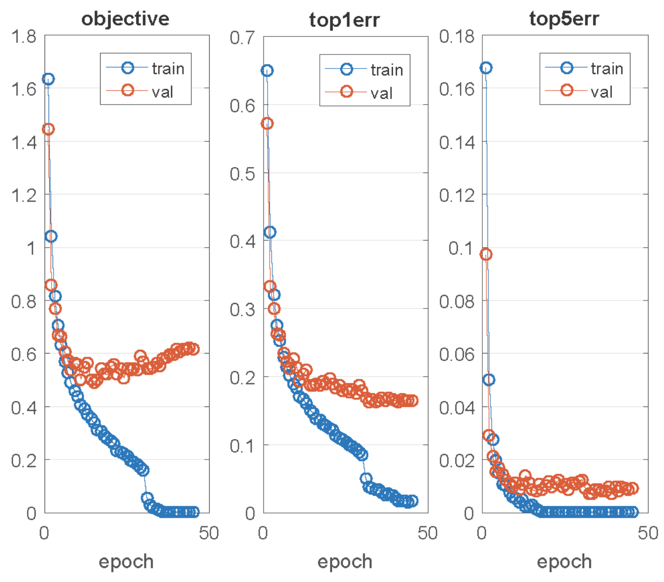

Figure 7.7: Multipartite pooling for CIFAR-10. Besides smaller losses and errors, with respect to the max pooling (Figure 7.6), the training (train) and test (val) performances indicate closer gaps, due to better generalization of the model, using by multipartite pooling.

Chapter 8

Distributed Backpropagation for Transfer Learning

Transfer learning is a popular practice in deep neural networks. Fine-tuning of a large number of parameters, is difficult due to the complex wiring of neurons between splitting layers. The imbalanced distributions of data for primary and target domains, also adds up to the hurdle of the problem. The reconstruction of the primary wiring for the target network is a heavy burden, considering the size of interconnections across neurons.

For supervised learning, many classification algorithms assume the same distributions for training and test data. In order to change this distribution, the statistical models must be rebuilt. This is not always practical, due to the difficulty of recollecting the training data or complexity of the learning process. One of the solutions is transfer learning, which transfers the classification knowledge into a new domain (Pan and Yang [2010]). The aim is to learn highly generalized models for either domains with different probability distributions, or domains without labelled data (Wang and Schneider [2014], Zhang et al. [2013]). Here, the main challenge is to reduce the shifts in data distributions between the domains, by using algorithms that minimize their discrimination. It is worth mentioning that, this cannot resolve domain-specific variations (Long et al. [2016]).

Transfer learning has also proven to be highly beneficial at boosting the overall performance of deep neural networks. Deep learning practices usually require a huge amount of labelled data to learn powerful models. The transfer learning enables adaptation to a different source with a small number of training samples.

On the other hand, deep neural networks practically learn intermediate features. They can provide better transfer learning, because some of them generalize well, among various domains of knowledge (Glorot et al. [2011]). These transferable features generally underlie several probability distributions (Oquab et al. [2014]), to reduce the cross-domain discrepancy (Yosinski et al. [2014]). The common observation among several deep architectures, is that, features learned in the bottom layers are not that specific, but transiting towards top layers, tailors them to the target dataset or task.

A recent study of the generality or specificity of deep layers for transfer learning, show two difficulties which may affect the transfer of deep features (Yosinski et al. [2014]). First, top layers get quite specialized to their primary tasks and second, some optimization difficulties arise due to the splitting of the network between adapted layers. In spite of these negative effects, other studies have confirmed that transferred features, not only perform better than random representation, but also provide better initialization.

This chapter proposes a distributed backpropagation scheme in which, the convolutional filters are fine-tuned individually, but are backpropagated, all at once. This is done by means of Basic Probability Assignment (BPA) (Sentz and Ferson [2002]) of evidence theory. Therefore, the primary filters are gradually transferred, based on, their contributions to classification performance of the target domain. This approach largely reduces the complexity of transfer learning, whilst improving precision. The experimental results on standard benchmarks and various scenarios, confirm the consistent improvement of the distributed backpropagation strategy for the transfer learning.

8.1 Method

A novel framework for distributed backpropagation in deep convolutional networks is introduced, that alleviates the burden of splitting a network through the middle of fragile layers. The intuition is that, this difficulty relates to the complexity of deep architecture and the imbalance data distribution of the primary and target domains.

On one hand, the splitting of layers, results in optimization difficulty, because there is a high complexity in the interconnections between neurons of adapted layers. To address this, the convolutional filters are fine-tuned individually. This reduces the complexity of non-convex optimization for the transfer learning problem. On the other hand, the imbalance problem arises form different distributions of data, in the primary and target domains. This issue can be handled by cost-sensitive, imbalanced learning methods. Since the power of deep neural models, comes from mutual optimization of all parameters, the above fine-tuned convolutional filters are joined by a cost-sensitive backpropagation scheme.

The emergence of new cost-sensitive methods for imbalanced data (Elkan [2001]), enables the misclassification costs to be embedded in the form of a cost matrix. Meaningful information is then able to be distributed to the learning process. The error, based on the misclassification costs, is measured for each class, to form a confusion matrix. This matrix is the most informative contingency table in imbalanced learning problems, because it gives the success rate of a classifier in a special class, and the failure rate on distinguishing that class from other classes. The confusion matrix has proven to be a great regularizer; smoothing the accuracy among imbalanced data and giving importance to minority distributions (Ralaivola [2012]).

Determination of a probabilistic distribution from the confusion matrix, is highly effective at producing a probability assignment, which contributes to the imbalanced problems. This can be either constructed from recognition, substitution and rejection rates (Xu et al. [1992]) or both precision and recall rates (Deng et al. [2016]). The key point is to harvest maximum possible prior knowledge from the confusion matrix, to overcome an imbalanced challenge. The experiments confirm the advantage of the proposed distributed backpropagation for transfer learning in deep convolutional neural networks.

8.2 Formulation

It is a general practice in transfer learning to include training of a primary deep neural network on a dataset, followed by fine-tuning of learned features for another dataset, on a new target network (Bengio et al. [2012]). The generality of selected features for both of the primary and target domains, is critical to the success of transfer learning.

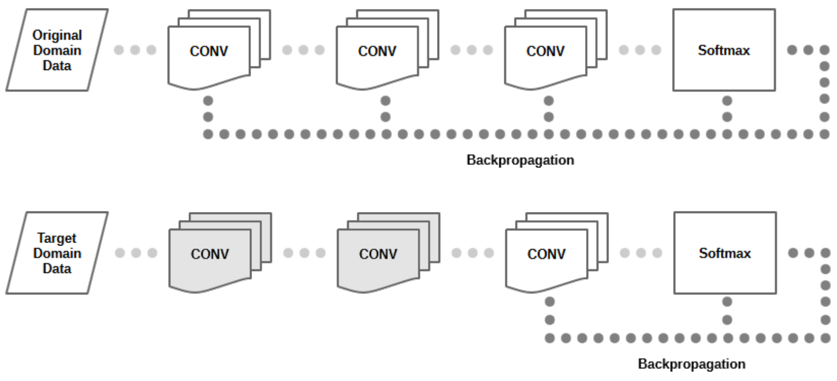

Figure 8.1: Standard backpropagation for transfer learning. The parameters of the target network can be either updated, in all layers or be frozen in the bottom layers, during backpropagation. Overfitting is highly probable for a small target dataset on a large primary network, due to the number of parameters. In contrast, underfitting is possible for a large target dataset on a small primary network, because of imbalanced data distributions. To fine-tune the pretrained network (primary) for the transferred network (target), backpropagation is applied to the top layers of the deep network, fed by the target data, while bottom layers remain fixed.

For implementation, the primary network is trained and its bottom layers copied, to form the target network. The top layers of the target network, are initialized randomly and are trained by the target data. It is possible to employ backpropagation for all layers and fine-tune their parameters for the target task, or freeze the copied primary layers, and only update the top target layers. This is usually decided by the size of the target dataset and number of parameters in the primary layers. Fine-tuning of large networks for small datasets, leads to overfitting, but for small networks and large datasets, improves the performance (Sermanet et al. [2013]). A diagram of standard backpropagation for transfer learning is presented in Figure 8.1.

To handle the overfitting issues, a distributed backpropagation paradigm is proposed. First, the large number of fine-tuning parameters are divided, so as, to conquer the complexity of the primary non-convex optimization. This is implemented by breaking of the l-layer network, of depth d, to the d distributed l-layer single-filter networks, of depth one. This leads to a decay rate of $1/d$ in the number of parameters, needing to be fine-tuned in each single-filter networks. Second, this distributed architecture is fed with the target data. As a result, each of the single-filter networks, generates the contingency matrix of its Softmax classifier for the expected classification, recognition, or regression task.

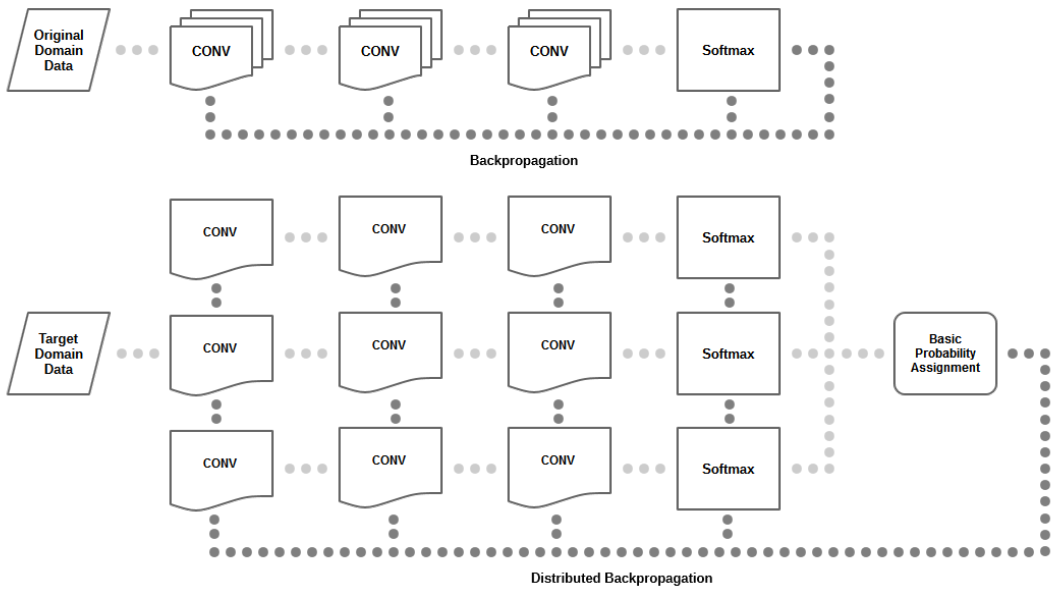

Figure 8.2: Distributed backpropagation for transfer learning. The primary network, pretrained on the primary domain, is arranged in several single-filter networks, each of which fine-tunes a single convolutional filter. For backpropagation, BPA is employed to determine the contribution of each single-filter networks. The computed costs are multiplied by the error gradients to backpropagate through the distributed single-filter networks, which are now fed by the target data. The distributed backpropagation controls the rate of change in weights and biases by a cost-sensitive formulation. As a result, the primary convolutional filters contribute significantly to the performance of the target domain, generating larger steps for gradient descent optimization. In contrast, the primary filters with small probability assignments, stop iterative optimization by causing infinitesimal steps. This leads to a powerful transferred model, which is better tailored to the target data distribution.

Third, BPA is employed to find out the contribution of each single-filter networks, to the specific learning task, in the target domain. Finally, the calculated probability assignments are normalized, to use as the costs of backpropagation, in each of the single-filter networks. In other words, the parameters are updated by the multiplication of error gradients, and the cost of each distributed single-filter networks (Figure 8.2).

The single-filter architectures are initialized by parameters, learned from the primary domain, meanwhile, they are optimized by gradient descent through backpropagation on the target domain. It differs from ensemble learning in that, single-filter networks are not re-weighted, but their parameters are updated to cope with the target domain.

8.2.1 Basic Probability Assignment

A confusion matrix is generally represented as class-based predictions against actual labels, in the form of a square matrix. Inspired by Dempster-Schafer theory (Sentz and Ferson [2002]), construction of BPA gives a vector, which is independent of the number of samples in each classes and sums up to one, for each labels. BPA provides the ability to reflect the different contributions of a classifier, or combine the outcomes of multiple weak classifiers. A raw, two-dimensional confusion matrix, indexed by the predicted classes and actual labels, provides some common measures of classification performance. Some general measures are accuracy (the proportion of the total number of predictions that are correct), precision (a measure of the accuracy, provided that a specific class has been predicted), recall (a measure of the ability of a prediction model to select instances of a certain class from a dataset), and F-score (the harmonic mean of precision and recall) (Sammut and Webb [2011]).

Suppose that a set of training samples $\mathcal{X} = \{X_1, \ldots, X_{|\mathcal{X}|}\}$ from $\mathcal{C} = \{C_1, \ldots, C_{|\mathcal{C}|}\}$ different classes, are assigned to a label set $\mathcal{L} = \{L_1, \ldots, L_{|\mathcal{L}|}\}$, using a classifier ϕ. Each element n_{ij} of the confusion matrix $\mathbf{Y}_\phi$ is considered as the number of samples belonging to class C_i, which assigned to label L_j. The recall (r_{ij}) and precision (s_{ij}) ratios for all the $i \in \{1 \ldots |\mathcal{C}|\}$ and $j \in \{1 \ldots |\mathcal{L}|\}$, can be defined as follows (Deng et al. [2016]),

$$\begin{aligned} r_{ij} &= \frac{n_{ij}}{\sum_{i=1}^{|\mathcal{C}|} n_{ij}} \\ s_{ij} &= \frac{n_{ij}}{\sum_{j=1}^{|\mathcal{L}|} n_{ij}} \end{aligned} \tag{8.1}$$

It can be seen that, the recall ratio is summed over the predicted classes (rows), whilst the precision ratio is accumulated by the actual labels (columns) of the confusion matrix $\mathbf{Y}_\phi$. The probability elements of recall (R_i) and precision (S_i) for each individual class C_i are,

$$\begin{aligned} R_i &= \frac{r_{ii}}{\sum_{j=1}^{|\mathcal{C}|} r_{jj}} \\ S_i &= \frac{s_{ii}}{\sum_{j=1}^{|\mathcal{L}|} s_{jj}} \end{aligned} \tag{8.2}$$

Algorithm 7 Basic Probability Assignment

```
Input: training set $\mathcal{X}$, classifier $\phi$
Output: basic probability assignment $\boldsymbol{\Theta}_\phi$

1: Compute recall and precision (Equation 8.1)
2: Calculate recall and precision assignments (Equation 8.2)
3: Apply Dempster-Schafer rule of combination (Equation 8.3)
```

These elements are synthesized to form the final probability assignments by Dempster-Schafer rule of combination (Sentz and Ferson [2002]), representing the recognition ability of classifier ϕ to each of the classes of set $\mathcal{C}$ as,

$$M_i = R_i \oplus S_i = \frac{R_i S_i}{1 - \sum_{i=1}^{|\mathcal{C}|} R_i S_i} \tag{8.3}$$

where the operator $\oplus$ is an orthogonal sum. The overall contribution of the classifier ϕ can be presented as a probability assignment vector,

$$\boldsymbol{\Theta}_\phi = \{M_i \mid \forall\, i \in [1, |\mathcal{C}|]\} \tag{8.4}$$

It is worth mentioning that $\boldsymbol{\Theta}_\phi$ should be computed by the training set, because it is assumed that, there is no actual label set $\mathcal{L}$ at the test time.

8.2.2 Distributed Backpropagation

Suppose that $\Phi = \{\phi_1, \ldots, \phi_{|\Phi|}\}$ is a set of Softmax loss functions of the single-filter networks, presented in Figure 8.2. To apply the distributed backpropagation, Algorithm 7 should be followed for each of the classifiers. The result is a set of normalized probability assignments as follows,

$$\boldsymbol{\Gamma}_\Phi = \{\boldsymbol{\Gamma}_{\phi_k} = \|\boldsymbol{\Theta}_{\phi_k}\|_2 \mid \forall\, k \in [1, |\Phi|]\} \tag{8.5}$$

Algorithm 8 Distributed Backpropagation for Transfer Learning

```
Input: primary network, target data
Output: fine-tuned target network

1: Compute $\mathbf{\Gamma}_\Phi$ (Equation 8.5) by using Algorithm 7

for $k = 1$ to $|\Phi|$ do
    for all layers do
        2: Compute feedforward activations (Equation 8.6)
        3: Calculate backpropagation errors (Equation 8.8)
        4: Update weights and biases (Equation 8.9)
    end for
end for
```

It is known that in each layer l of the kth single-filter network, the feedforward propagation is calculated as,

$$a^{(l)} = \sigma(w^{(l)}a^{(l-1)} + b^{(l)}) \tag{8.6}$$

which w and b are weights and biases, a is an activation and σ is a rectification function. Considering ϕ as the cost function of the kth network, the output error holds,

$$\delta^{(l)} = \bigtriangledown_a \phi \odot \sigma^{'}(w^{(l)}a^{(l-1)} + b^{(l)}) \tag{8.7}$$

and the backpropagation error can be stated as,

$$\delta^{(l)} = ((w^{(l+1)})^T)\delta^{(l+1)} \odot \sigma^{'}(w^{(l)}a^{(l-1)} + b^{(l)}) \tag{8.8}$$

For the sake of gradient descent, the weights and biases are updated via,

$$\begin{aligned} w^{(1)} &\longrightarrow w^{(1)} - \eta\ \Gamma_\phi \sum \delta^{(l+1)}(a^{(l-1)})^T \\ b^{(1)} &\longrightarrow b^{(1)} - \eta\ \Gamma_\phi \sum \delta^{(l+1)} \end{aligned} \tag{8.9}$$

It can be seen that, greater $\mathbf{\Gamma}_\phi$ of the kth single-filter network, makes larger steps to update the weights and biases of the target domain, during distributed backpropagation, compared to the standard backpropagation, with a fix η in the primary domain. This helps to only update primary filters, which largely affect the target domain and also properly connect the distributed single-filter networks, during the fine-tuning process. This also implies that, in spite of forward-backward propagation in the target domain, the overall contribution of all convolutional filters, is taken into account. Since the united backpropagation is involved in every iterations of training, the single-filter networks are trained together, which is different from ensemble learning. Algorithm 8 wraps up the proposed distributed strategy.

It is assumed that the number of classes and assigned labels are equal ($|\mathcal{C}| = |\mathcal{L}|$), although the merging of different classes is a common practice, particularly in visual classification (for example, vertical vs horizontal categories). The benefit lies in the fact that the bottom layers of deep convolutional architectures, contribute to detecting first and second order features, that are usually of specific directions, rather than specific distinguishing patterns of the objects. This leads to a powerful hierarchical feature learning, in the case $|\mathcal{C}| \ll |\mathcal{L}|$. In contrast, some classes can be divided into various subcategories, although they all get the same initial labels, and hence this holds $|\mathcal{C}| \gg |\mathcal{L}|$ to take the advantage of more general features in the top layers. The proposed distributed backpropagation does not merge or divide the primary labels of the datasets under study, although it seems that, this is able to boost the performance of transfer learning, for both of the merging or dividing cases.

8.3 Experiments

Two different scenarios are considered to evaluate the performance of distributed backpropagation for transfer learning. In the first scenario, the performance of fine-tuning for pairs of datasets, with either close data distributions or number of classes, are observed. These are MNIST & SVHN and CIFAR-10 & CIFAR-100 pairs, as primary & target domains, and the performance of transfer learning, are reported in the form of training-test errors. For the second scenario, transfer learning is applied for pairs of datasets, with far data-class distributions, which are MNIST & CIFAR-10 and SVHN & CIFAR-100 pairs. In these experiments, the datasets are arranged to examine the effect of dissimilar distributions, rather than overfitting.

Dataset	Baseline	
	Train (%)	Test (%)
MNIST	0.04	0.55
SVHN	0.13	3.81
CIFAR-10	0.01	19.40
CIFAR-100	0.17	50.90

Table 8.1: Baselines of classification errors on standard benchmarks. The results are produced by deep learning of convolutional neural networks, on the datasets under examination.

As a practical example, suppose that the aim is to transfer MNIST as the primary domain, to CIFAR as the target domain. To initialize, CIFAR data is presented to the MNIST pretrained network, filter by filter, for each of the 20 convolutional filters of the MNIST network. Then, the outputs of Softmax layers are passed to BPA module, and the error gradients are multiplied by the specific cost of each filters. These new gradients backpropagate to all 20 single-filter networks, to update their weights and biases. After some iterations, the MNIST network is transferred to the CIFAR network, through the distributed backpropagation scheme, running on 20 fine-tuned, single-filter networks. The baselines of training and test errors on the experimental datasets, are reported in Table 8.1. For ease of implementation and fast replication of the results, the deep learning library provided by the Oxford Visual Geometry Group (Vedaldi and Fulkerson [2008]), is deployed.

8.3.1 Transfer Learning on Fairly Balanced Domains

In this scenario, two pairs of datasets are targeted, which contain similar data distributions and perform the same recognition tasks. The results are reported for standard vs distributed backpropagations, in Table 8.2. The standard backpropagation trains the primary network and fine-tunes the top layers for the target network (Bengio et al. [2012]). The proposed distributed backpropagation employs BPA, for the cost-sensitive transfer learning.

It can be seen that, the results for the standard backpropagation follow the argument on size of networks and number of model parameters (Sermanet et al. [2013]). MNIST does a poor job on transferring to SVHN, due to the overfitting of SVHN over MNIST network. In contrast, SVHN performs well as the primary network to transfer MNIST as the target domain.

primary	Target	Standard		Distributed	
		Train (%)	Test (%)	Train (%)	Test (%)
MNIST	SVHN	**0.01**	29.57	0.24	**5.18**
SVHN	MNIST	**0.35**	1.04	0.16	**0.46**
CIFAR-10	CIFAR-100	0.53	68.44	**0.29**	**54.32**
CIFAR-100	CIFAR-10	0.11	24.08	**0.05**	**18.24**

Table 8.2: Classification errors of standard vs distributed backpropagations, for transfer learning on fairly balanced domains. Bold values correspond to better performances, which are the smallest training and test errors on each experiment.

On the other hand, transferring to SVHN domain from MNIST, does not result in overfitting, when the distributed backpropagation is employed. In both settings of the primary and target domains, the distributed strategy outperforms the standard backpropagation.

The experiments based on the CIFAR pair, raise interesting results, because both datasets have the same number of samples, but completely different distributions among the classes. In practice, CIFAR-100 includes all the classes of CIFAR-10, but CIFAR-10 is not aware of several classes in CIFAR-100. It can be seen that, CIFAR-10 transfers well to CIFAR-100. This cannot outperform the baseline performance, although the target network (CIFAR-100) is not overfitted. All in all, the performance of the distributed backpropagation for transfer learning is better than the standard scheme and also, outperforms the baselines of the benchmarks.

8.3.2 Transfer Learning on Highly Imbalanced Domains

This scenario pairs the datasets, such that, the similarity of their data distributions, and the number of classes get minimized. They are also initially trained for different tasks, *i.e.* number classification vs object recognition. For implementation, MNIST gray channel is repeated three times to make a RGB-like colour representation for transferring to CIFAR network. For the CIFAR, the RGB channels are converted into grayscale to transfer on the MNIST network.

From Table 8.3, it is obvious that the distributed backpropagation outperforms all the standard results. For the first setup, CIFAR-10 performs better at transfer learning than MNSIT, although the number of classes are the same.

primary	Target	Standard		Distributed	
		Train (%)	Test (%)	Train (%)	Test (%)
MNIST	CIFAR-10	0.43	28.92	**0.25**	**20.85**
CIFAR-10	MNIST	0.44	2.37	**0.23**	**0.95**
SVHN	CIFAR-100	0.71	89.31	**0.46**	**61.10**
CIFAR-100	SVHN	**0.01**	12.18	0.28	**7.25**

Table 8.3: Classification errors of standard vs distributed backpropagations, for transfer learning on highly imbalanced domains. Bold values correspond to better performances, which are the smallest training and test errors on each experiment.

It seems that, CIFAR-10 provides better generalization due to higher diversity among its classes. Here, the distributed backpropagation performs better than the standard process and, targeting of MNIST from CIFAR-10 network, results in a performance that is similar to the baseline outcomes on MNIST in Table 8.1. The second setup leads to the overfitting of SVHN over CIFAR-100 network, as a result of the large number of samples. The other outcome is the poor performance of transferring CIFAR-100 to SVHN network. This is the result of very different contents of primary and target datasets.

The observations show that fine-tuning on the training set, while calculating BPA on the validation set, result in better generalization of the transferred model. On the other hand, computing of BPA on training plus validation sets, gives a higher performance. This is due to the vastly different number of classes in the primary-target domains. Since BPA is employed to address the imbalance distribution problem, it better captures the distribution of data, by adjoining both training and validation sets. This is especially true, when fewer classes of the primary domain, are transferred to the larger number of classes in the target domain.

8.4 Conclusion

In this chapter, a distributed transfer learning algorithm for deep convolutional neural networks, was proposed. Since fine-tuning of large networks for small datasets mostly led to overfitting, this provided less susceptibility by breaking the network to single-filter networks of the same depth as primary network. This was also fed with the target data and BPA was employed to find the contribution of each single-filter network to the particular task. The experiments confirmed that the proposed framework improved the accuracy of transfer learning.

Chapter 9

Unified Backpropagation for Multi-Objective Learning

A common practice in most deep convolutional neural networks, is to employ fully-connected layers, followed by a Softmax activation to minimize cross-entropy loss. Recent studies has shown that, substitution of the Softmax objective with SVM or LDA cost functions, is highly effective to improve the classification performance of deep neural networks. This chapter proposes a novel paradigm to link the optimization of several objectives through a unified backpropagation scheme. This alleviates the burden of extensive boosting for each independent objective functions and avoids complex formulation of multi-objective gradients. Here, several loss functions are linked through BPA at the time of backpropagation.

Deep learning has been proven to be extremely successful for several applications. The combination of machine learning methods, with deep neural networks, achieves better performances. Deep versions of CCA (Andrew et al. [2013]), FA (Clevert et al. [2015]), PCA (Chan et al. [2015]), SVM (Vinyals et al. [2012]), and finally, LDA (Stuhlsatz et al. [2012]), have been introduced in the literature. There are two schools of thought about how to alternate the Softmax layer, so as, to achieve better performance.

The first strategy trains a deep architecture to produce high-order features and give them to a classifier (Coates et al. [2010]). For example, replacing the Softmax with SVM as the top layer and minimizing of a standard hinge loss, produces better results in some deep architectures (Tang [2013]). Another successful practice is LDA which maximizes an objective, derived from a general eigenvalue problem (Dorfer et al. [2015]). The drawback is that, the features in the bottom layers are not further fine-tuned with the new objective.

The second strategy trains a combination of objectives, by error backpropagation through the gradients of their loss functions. It is possible to optimize these objectives with either boosting methods or multi-objective evolutionary algorithms (Gong et al. [2015]), but the former needs extensive training of different networks, and the latter requires very complex formulations for the gradients.

9.1 Method

A novel unified backpropagation scheme is proposed for deep multi-objective learning, based on BPA of evidence theory, applying to a network that includes different loss functions, such as, Softmax, SVM, and LDA. In contrast to the boosting method, which trains each loss function independently to make an ensemble of models, the proposed backpropagation approach, unifies the gradients of all objective functions.

The advantages of unified backpropagation can be outlined as follows. First, this scheme mutually optimizes all the objective functions together. In this way, the contribution of each objective function to the overall classification performance, is managed by sharing the basic probability masses, among the gradients. Second, this unification is less computationally expensive than ensemble learning. Third, it prevents more complexity on the formulation of gradients, for each of the combined loss functions. The experiments for a variety of scenarios and standard datasets, confirm the advantage of the proposed approach, which delivers consistent improvements to the performance of deep convolutional neural networks.

9.2 Formulation

A deep convolutional architecture can boost the typical Softmax layer with other classifiers, using a multi-objective optimization regime. The two widely-used classifiers are SVM and LDA, employed as the top layer of the deep neural networks.

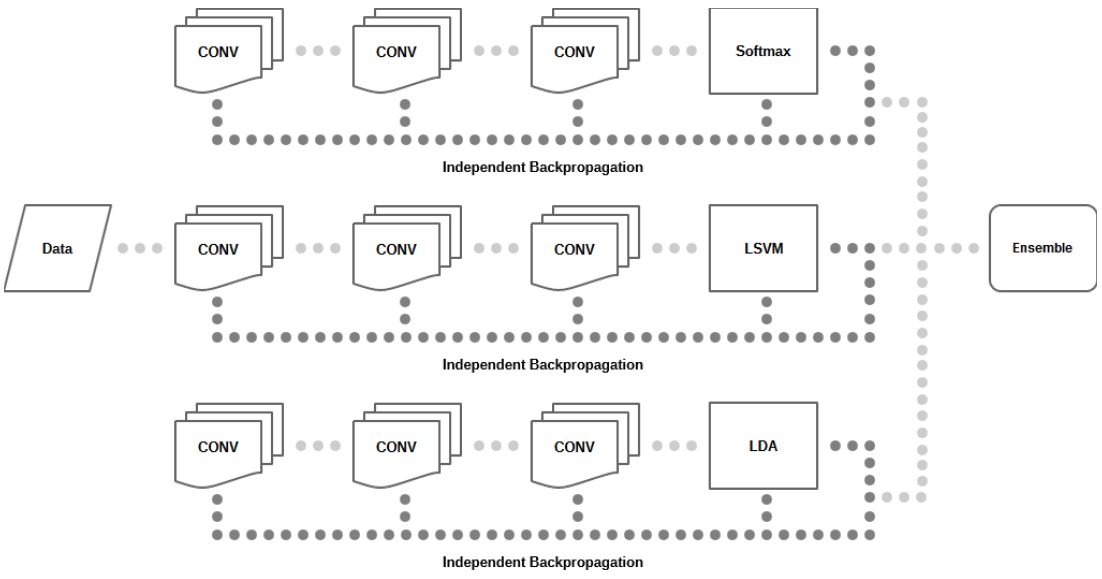
CONV
CONV
CONV
Softmax
Independent Backpropagation
Data
CONV
CONV
CONV
LSVM
Ensemble
Independent Backpropagation
CONV
CONV
CONV
LDA
Independent Backpropagation

Figure 9.1: Multi-objective learning of deep convolutional neural networks. This follows the common practice of backpropagation in identical convolutional architectures and minimizing some specific loss functions, via stochastic gradient descent. The trained deep classifiers are then boosted by a proper ensemble method. The bottleneck results from the need for multiple instances of training for each of the objective functions. The backpropagation procedures are thoroughly independent, because each of the loss functions minimize by a specific gradient formulation, with zero awareness of other ongoing training processes.

Employing SVM, a deep convolutional network optimizes the primal problem of SVM and backpropagate the gradients of the top layer SVM, to learn the bottom layers. This is in contrast with fine-tuning, where low-order features are usually trained by Softmax, and top layers tuned by SVM. It has been shown that the performance on standard benchmarks, is much better than the networks with Softmax layer at top. The optimization is performed using stochastic gradient descent method (Chan et al. [2015]).

For LDA as the top layer, the deep architecture is almost the same. The objective of a deep neural network is reformulated to learn linearly separable features by backpropagation, because LDA allows to define optimal linear decision boundaries in the latent layers. This finds linear combinations of low-level features, to maximize the scattering between classes of data, whilst minimizing the discrepancy within individual classes. The top layer LDA tries to produce high separation between deep features, rather than, minimizing the norm of prediction error (Dorfer et al. [2015]).

Classification of some multiclass datasets, is challenging due to non-uniform distribution of data (Koço and Capponi [2013]). The accuracy of classification does not seem to be a suitable objective to optimize, because there may be high accuracy with strong biases towards some classes (Fawcett [2006]). Although there are many algorithms to deal with imbalanced data for binary classification (He and Garcia [2009]), the multiclass problem has been usually addressed by generalization of the binary solutions, with a one-versus-all strategy (Abe et al. [2004]). For some leaning tasks, optimization of some relevant measures within the imbalance data distribution, provides alternative measures to the accuracy (Wang and Yao [2012]).

The emergence of new cost-sensitive methods for dealing with imbalanced multiclass data (Elkan [2001]), has enabled the embedding of misclassification costs, into a cost matrix. They usually measure the error, based on misclassification costs of individual classes in the confusion matrix. This matrix is the most informative contingency table in multiclass learning problems, because it gives the success rate of a classifier for a special class, and the failure rate on distinguishing that class from other classes. The confusion matrix has proven to be a great regularizer; smoothing the accuracy among classes (Ralaivola [2012]).

The determination of a probabilistic distribution from the confusion matrix is highly effective at producing a probability assignment, which contributes to imbalance distribution problems. The probability assignments can be constructed from recognition, substitution and rejection rates (Xu et al. [1992]), or both precision and recall rates (Deng et al. [2016]). The key point is to harvest the maximum possible prior knowledge, provided by the confusion matrix to overcome the imbalance classification challenge.

9.2.1 Unified Backpropagation

Suppose that $\Phi = \{\phi_1, \ldots, \phi_{|\Phi|}\}$ is a set of objectives in a multi-objective learning regime, presented in Figure 9.2. To apply the unified backpropagation, Algorithm 7 is deployed for each of the loss functions, to come up with a set of normalized probability assignments as,

$$\mathbf{\Gamma}_\Phi = \{\mathbf{\Gamma}_{\phi_k} = \|\mathbf{\Theta}_{\phi_k}\|_2 \mid \forall\, k \in [1, |\Phi|]\} \tag{9.1}$$

where $\mathbf{\Theta}_{\phi_k}$ follows the same definition as Equation 8.4.

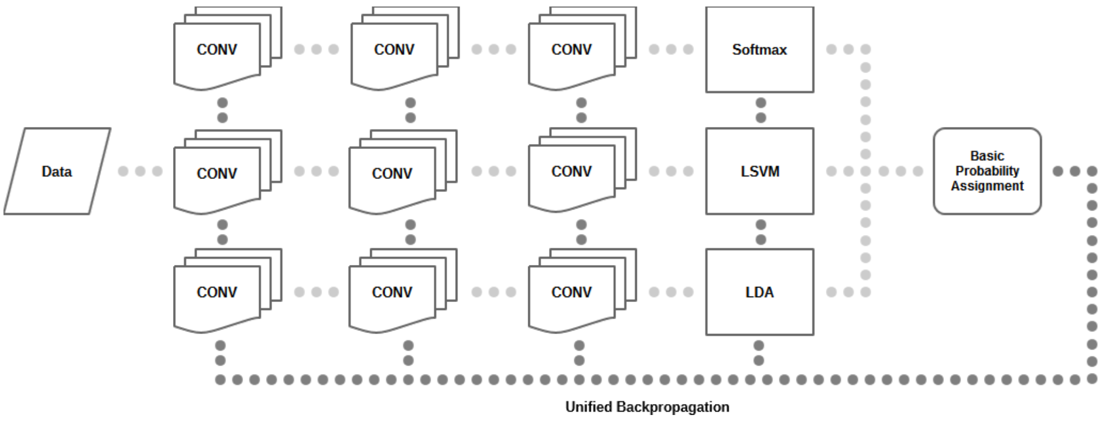

Figure 9.2: Unified backpropagation for multi-objective learning. In contrast with the ensemble practice, the backpropagation is unified through BPA. Although each deep convolutional network holds its specific gradient formulation, sharing of probability masses, makes a common sense around the statistics of gradient descent in the whole network. In other words, boosting procedure not only includes the outcome of several objective functions, but also enhances their mutual learning processes, by unifying their backpropagation.

In each layer l of the kth objective, feedforward propagation is calculated as follows,

$$a_k^{(l)} = \sigma(w_k^{(l)} a_k^{(l-1)} + b_k^{(l)}) \tag{9.2}$$

where w_k and b_k are weights and biases, a_k is an activation and σ is a rectification function. Considering ϕ_k as the loss function, the output error holds,

$$\delta_k^{(l)} = \bigtriangledown_{a_k}\phi_k \odot \sigma'(w_k^{(l)} a_k^{(l-1)} + b_k^{(l)}) \tag{9.3}$$

The backpropagation error can be stated as,

$$\delta_k^{(l)} = ((w_k^{(l+1)})^T)\delta_k^{(l+1)} \odot \sigma'(w_k^{(l)} a_k^{(l-1)} + b_k^{(l)}) \tag{9.4}$$

For the sake of gradient descent, the weights and biases are updated via,

$$\begin{aligned} w_k^{(1)} &\longrightarrow w_k^{(1)} - \eta\, \Gamma_{\phi_k} \sum \delta_k^{(l+1)} (a_k^{(l-1)})^T \\ b_k^{(1)} &\longrightarrow b_k^{(1)} - \eta\, \Gamma_{\phi_k} \sum \delta_k^{(l+1)} \end{aligned} \tag{9.5}$$

Algorithm 9 Unified Backpropagation

```
Input: set of objective functions $\Phi$
Output: multi-objective network

1: Compute $\mathbf{\Gamma}_\Phi$ (Equation 9.1) by using Algorithm 7

for $k = 1$ to $|\Phi|$ do
  for all layers do
    2: Compute feadforward activations (Equation 9.2)
    3: Calculate backpropagation errors (Equation 9.4)
    4: Update weights and biases (Equation 9.5)
  end for
end for
```

For the unified backpropagation, larger Γ_{ϕ_k} of the kth objective will generate bigger update rates for weights and biases, than only employing a fix η. This helps to update only loss functions, which largely affect the overall classification performance, and properly connect the objectives in the backpropagation process. This also implies that, in spite of forward-backward propagation, the overall contribution of each objective function is taken into account. Algorithm 9 wraps up the proposed unification strategy.

9.2.2 Objective Functions

Following the successful practices in the literature, three types of widely-used loss functions *i.e.* Softmax, SVM (Tang [2013]), and LDA (Dorfer et al. [2015]) are further investigated. Suppose that $\mathcal{C} = \{C_1, \ldots, C_{|\mathcal{C}|}\}$ is a set of $|\mathcal{C}|$ different classes in the dataset at hand and the discrete probability distribution p_i denotes, to what extent, each sample of set $\mathcal{X} = \{X_1, \ldots, X_{|\mathcal{X}|}\}$ belongs to class C_i. Assuming $a_k = \{a_{k_1}, \ldots, a_{k_{|\mathcal{C}|}}\}$ as the output of the last fully-connected layer for the kth loss function (Equation 9.2), the closed form formulation of gradients for the above objective functions, can be worked out as follows.

9.2.2.1 Softmax

For a conventional Softmax activation, p_i can be defined as follows,

$$p_i = \frac{e^{a_{k_i}}}{\sum_{j=1}^{|\mathcal{C}|} e^{a_{k_j}}} \tag{9.6}$$

such that $\sum p_i = 1$ and the predicted class is yielded by,

$$y_i = \arg\max \; p_i = \arg\max \; a_i \tag{9.7}$$

The Softmax loss function forms as,

$$\phi_{k_{Softmax}} = -\sum_{j=1}^{|\mathcal{C}|} y_j \; log(p_j) \tag{9.8}$$

where its gradient with respect to a_{k_i} holds,

$$\bigtriangledown_{a_{k_i}} \phi_{k_{Softmax}} = p_i - y_i \tag{9.9}$$

9.2.2.2 Support Vector Machine

The squared hinge loss for L_2-norm binary SVM ($t_i \in \{-1, +1\}$) is defined as,

$$\phi_{k_{SVM}} = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \lambda \sum_{i=1}^{|\mathcal{C}|} max(0, 1 - \boldsymbol{w}^T a_{k_i} t_i)^2 \tag{9.10}$$

that its gradient can be derived as follows,

$$\bigtriangledown_{a_{k_i}} \phi_{k_{SVM}} = -2\lambda t_i \boldsymbol{w}\big(max(0, 1 - \boldsymbol{w}^T a_{k_i} t_i)\big) \tag{9.11}$$

The multiclass scenario is the extension of the binary objective, using one-vs-rest approach. Minimizing Equation 9.11 for $\boldsymbol{w}$, gives the predicted class as,

$$y_i = \arg\max \; w_i a_{k_i} \quad \forall \; w_i \in \boldsymbol{w} \tag{9.12}$$

9.2.2.3 Linear Discriminant Analysis

The focus is on maximizing the $|\mathcal{C}| - 1$ smallest eigenvalues of the generalized LDA eigenvalue problem,

$$S_b \boldsymbol{e} = \boldsymbol{v} S_w \boldsymbol{e} \tag{9.13}$$

such that, $\boldsymbol{S_b}$ is between-class scattering, $\boldsymbol{S_w}$ is within-class scattering, $\boldsymbol{e}$ corresponds to eigenvectors and $\boldsymbol{v}$ represents the eigenvalues. This leads to a maximization of the discriminant power of any deep architecture. Hence, the objective can be stated as,

$$\phi_{k_{LDA}} = \frac{1}{|\mathcal{C}| - 1} \sum_{i=1}^{|\mathcal{C}|-1} v_i \quad \forall\, v_i \in \boldsymbol{v} \tag{9.14}$$

which holds the following gradient ($\forall e_i \in \boldsymbol{e}$),

$$\nabla_{a_{k_i}} \phi_{k_{LDA}} = \frac{1}{|\mathcal{C}| - 1} \sum_{i=1}^{|\mathcal{C}|-1} {e_i}^T \left(\frac{\delta S_b}{\delta a_{k_i}} - \frac{\delta S_w}{\delta a_{k_i}} \right) e_i \tag{9.15}$$

9.3 Experiments

The experiments are conducted for two different scenarios, using MNIST (LeCun et al. [1998]), CIFAR (Krizhevsky and Hinton [2009]), and SVHN (Netzer et al. [2011]) datasets. In the first scenario, the unified backpropagation is applied to single-objective learning and its performance is compared to the baseline of Softmax, SVM and LDA. For the second scenario, multi-objective learning are considered and multiple loss functions are combined via the unification backpropagation paradigm. We report the results for standard deep architectures, implemented by the Oxford Visual Geometry Group (Vedaldi and Fulkerson [2008]).

9.3.1 Single-Objective Learning

In this scenario, the advantage of the unified backpropagation (Unified) is validate for each of the individual objective functions under examination. We provide the outcomes for the Softmax, SVM, and LDA as common loss functions among deep convolutional networks.

Tables 9.1, 9.2, and 9.3 show the outcomes of this scenario. It can be seen that, the unified backpropagation is able to consistently improve the classification performance on deep convolutional networks, and provides smaller training-test errors. According to the Table 9.1, the unified backpropagation outperforms all the baselines. The greatest improvement belongs to CIFAR-10, which reduces the test error from 22.72% to 18.77%. The smallest improvement on the training error, comes from the same datasets. It seems that, in spite of larger training error, better generalization leads to considerable enhancement in the overall performance.

In Table 9.2, the best improvement in test error goes to CIFAR-100, which decreases from 49.11% to 39.76%. Although the number of classes are higher for other datasets, the unified backpropagation successfully avoids biases for SVM and hence, the overall performance is considerably improved. Looking at the Table 9.3, the best result is recorded for MNIST. The unification paradigm outperforms in both training and test errors, on all the experimental datasets. This is the result of distinction, imposed by LDA. Since LDA pushes the separation among classes, rather than the likelihood of predictions and labels (Softmax,SVM), the training-test errors reduce accordingly in all the experiments. This confirms that the proposed scheme is successful at providing better learning practices, compared to the typical methods.

Figures 9.3 and 9.4 present the comparative plots for MNIST and CIFAR-10, respectively. It is obvious that the unified backpropagation provides better generalization, and improves the training-test errors of the classification task. In Figure 9.3, the gap between training and validation errors, hugely reduces by the proposed unification method. It means that, this provides better generalization for the trained model. the energy of loss function for Softmax is lower than the proposed method. Although this might result in the better performance for the former, the latter outperforms, in regards, to the test errors. The reason lies in the capability of this method to deal with non-smooth decision boundaries of non-convex objectives in deep neural networks. This is a critical point, especially when the classes are highly correlated in the datasets, as they are for MNSIT or CIFAR-10.

Dataset	Softmax		Unified	
	Train (%)	Test (%)	Train (%)	Test (%)
MNIST	0.09	0.65	**0.04**	**0.32**
CIFAR-10	**1.32**	22.72	1.56	**18.77**
CIFAR-100	**0.17**	50.90	0.21	**48.01**
SVHN	0.13	3.81	**0.07**	**2.59**

Table 9.1: Classification errors for Softmax and the unified backpropagation. Bold values indicate the minimum training-test errors for each datasets. The proposed unification approach outperforms in test errors. Softmax produces greater training precisions for CIFAR datasets, than MNIST and SVHN.

Dataset	SVM		Unified	
	Train (%)	Test (%)	Train (%)	Test (%)
MNIST	**0.07**	0.53	0.08	**0.51**
CIFAR-10	**1.27**	19.87	1.47	**17.64**
CIFAR-100	**0.13**	49.11	0.17	**39.76**
SVHN	0.12	3.46	**0.09**	**2.38**

Table 9.2: Classification errors for SVM and the unified backpropagation. Bold values indicate the minimum training-test errors for each datasets. The unification paradigm gives the best test performances, but SVM outperforms on training errors for all datasets, except SVHN.

Dataset	LDA		Unified	
	Train (%)	Test (%)	Train (%)	Test (%)
MNIST	0.07	0.42	**0.05**	**0.38**
CIFAR-10	0.94	7.39	**0.87**	**6.95**
CIFAR-100	0.15	19.63	**0.13**	**18.46**
SVHN	0.06	2.78	**0.06**	**2.27**

Table 9.3: Classification errors for LDA and the unified backpropagation. Bold values indicate the minimum training-test errors for each datasets. The best performances come from the proposed unification scheme.

Figure 9.4 shows that the generalization for CIFAR-10, imposed by the unified backpropagation, is much higher than MNIST. As mentioned before, less correlation between classes in CIFAR-100 dataset, results in a better job at tuning the model parameters by this backpropagation strategy. Another observation for CIFAR-10 is that, the pattern of variations in the baseline and outcomes, is not highly aligned with MNIST. It seems that in some epochs, the unified backpropagation forces the learning process towards special classes, which do not contribute to the overall precision of classification.

9.3.2 Multi-Objective Learning

This scenario applies the unified backpropagation to combine Softmax, SVM, and LDA objective functions. The baselines are produced by ensemble learning via Adaboost algorithm. Tables 9.4, 9.5, and 9.6 summarize the training-test errors on all the experimental datasets, for Softmax + SVM, Softmax + LDA, and Softmax + SVM + LDA, powered by the proposed backpropagation paradigm. It can be seen that, almost everywhere, the proposed unification improves the classification performance. The only exception is Softmax + LDA on CIFAR-100 dataset, where this method is not able to outperform the baseline.

Table 9.4 shows the outcomes of the unification on Softmax and SVM combination. It is obvious that, the unified backpropagation improves test errors for all the experiments. The training errors are all improved, except on CIFAR-100 dataset. In table 9.5, we report the errors for the joint Softmax and LDA. Here, all the improvements come from the unified backpropagation. On CIFAR-100, training error increases from 0.9% to 1.5%, and on CIFAR-100 the testing error jumps from 18.27% to 35.28%. Since these degradations belong to CIFAR, it can be concluded that, LDA is not that successful at separating their highly-correlated classes.

Finally, Table 9.6 gathers the results of experiments for the composition of Softmax, SVM, and LDA. It is clear that the unification scheme outperforms the baseline on all training-test errors, except for the training on CIFAR-10 and test on CIFAR-100 datasets. It seems that SVM makes a significant contribution towards compensating LDA's disadvantage on CIFAR datasets, but that is not enough for the unified backpropagation to take an edge over baseline.

Dataset	Softmax + SVM		Unified	
	Train (%)	Test (%)	Train (%)	Test (%)
MNIST	0.08	0.57	**0.07**	**0.45**
CIFAR-10	1.54	18.72	**1.15**	**15.82**
CIFAR-100	**0.23**	48.85	0.91	**38.58**
SVHN	0.11	3.27	**0.08**	**2.48**

Table 9.4: Classification errors for Softmax + SVM and the unified backpropagation. Bold values indicate the minimum training-test errors for each datasets. The test errors show considerable improvements over the baseline performances.

Dataset	Softmax + LDA		Unified	
	Train (%)	Test (%)	Train (%)	Test (%)
MNIST	0.07	0.44	**0.05**	**0.41**
CIFAR-10	**0.90**	7.51	1.50	**6.81**
CIFAR-100	**0.18**	**18.27**	0.71	35.28
SVHN	0.09	3.64	**0.06**	**2.41**

Table 9.5: Classification errors for Softmax + LDA and the unified backpropagation. Bold values indicate the minimum training-test errors for each datasets. This approach improves the precision, when compared with the baseline for all datasets, except CIFAR-100.

Dataset	Softmax + SVM + LDA		Unified	
	Train (%)	Test (%)	Train (%)	Test (%)
MNIST	0.08	0.38	**0.05**	**0.30**
CIFAR-10	**0.78**	5.96	1.83	**5.44**
CIFAR-100	1.35	**22.49**	**0.68**	35.13
SVHN	0.08	3.01	**0.07**	**2.34**

Table 9.6: Classification errors for Softmax + SVM + LDA and the unified backpropagation. Bold values indicate the minimum training-test errors for each datasets. This paradigm outperforms the baseline on almost all the experimental datasets, except CIFAR-100.

Dataset	Baseline		Unified	
	Train (%)	Test (%)	Train (%)	Test (%)
MNIST	0.08	0.38	**0.05**	**0.30**
CIFAR-10	**0.94**	7.39	1.83	**5.44**
CIFAR-100	0.18	**18.27**	**0.13**	18.46
SVHN	0.07	2.78	**0.06**	**2.27**

Table 9.7: Minimum test errors for the baseline and unified backpropagation. Bold values indicate the minimum training-test errors for each datasets. The unification strategy is consistent at improving the classification performance of deep convolutional networks.

All in all, LDA does a better job than SVM in both of the baseline and unified backpropagation, and the proposed method performs better, when all the objectives come together. The best improvement goes to CIFAR-100 dataset, which reduces the test error from 38.58% for Softmax + SVM to 35.28% for Softmax + LDA, followed by 35.13% for Softmax + SVM + LDA. For CIFAR-10, Softmax + LDA improves the performance quite well, in comparison with, Softmax + SVM. Although the joint venture of all classifiers, generates higher precisions in test, the lowest training errors, varies between CIFAR-10 for Softmax + SVM, MNIST and SVHN for Softmax + LDA, and CIFAR-100 for Softmax + SVM + LDA.

9.3.3 Discussion

Considering both of the experimental scenarios, Table 9.7 summarizes the minimum test errors, and its corresponding training errors for each of the datasets under examination. The unified backpropagation either outperforms baselines by high margins (5.44% vs 7.39% for CIFAR-10) or follows them by close rates (18.46% vs 18.27% for CIFAR-100). This confirms advantage of the proposed backpropagation for the classification.

On the other hand, the best results for the unification method, go to multi-objective learning on MNIST & CIFAR-100 and single-objective learning on CIFAR-100 & SVHN datasets. Although further investigations remain, initial results indicate that multi-objective learning performs better on a small or medium number of samples-classes, while single-objective learning performs best on a large number of samples-classes. This is due to the fact that the multi-objective regime is not able to cope, with either complex data distributions, or highly-correlated classes, when several objectives contradict each other.

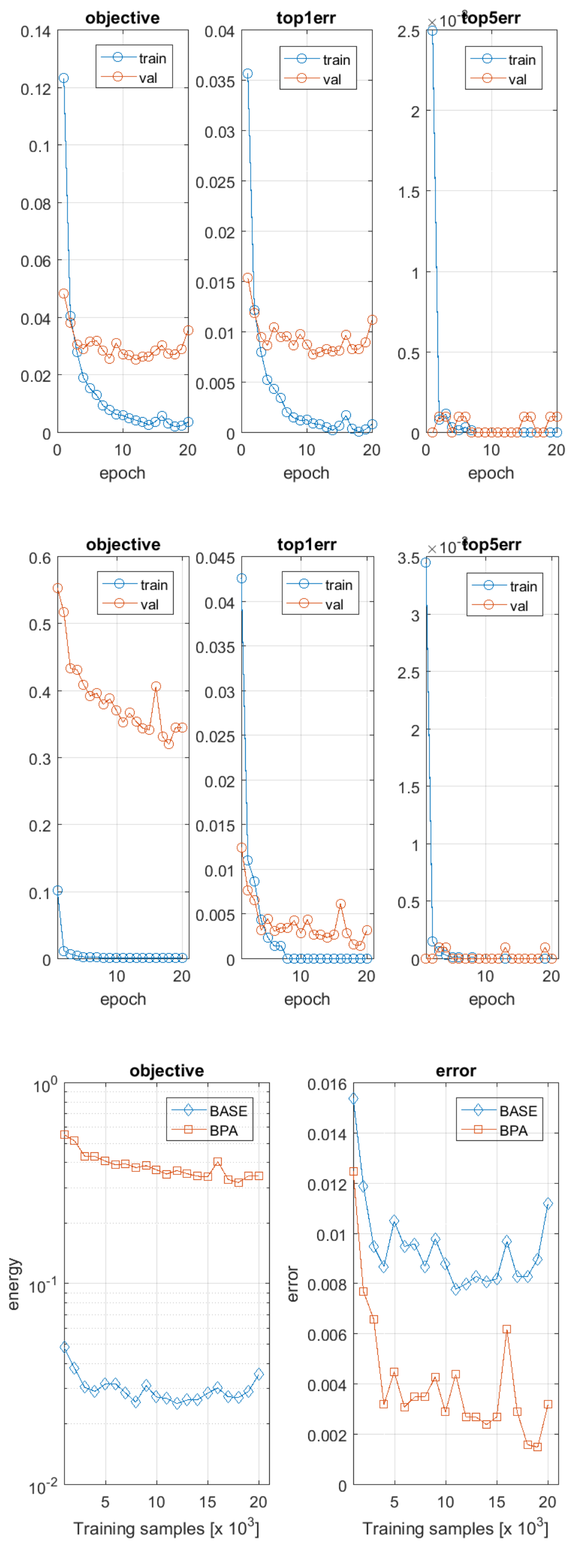

Figure 9.3: Unified backpropagation for MNIST dataset. For Softmax baseline (chart 1), the training-validation losses and top-1 errors, follow each other accordingly, but the gap between them is fairly wide. By the unification approach (chart2), the validation gives higher energy than the training, but the errors are considerably smaller than the baseline. This also provides better generalization, due to the closer gap between training-validation errors. In spite of the higher level of energy (chart 3), the unified backpropagation consistently improves the precision of Softmax.

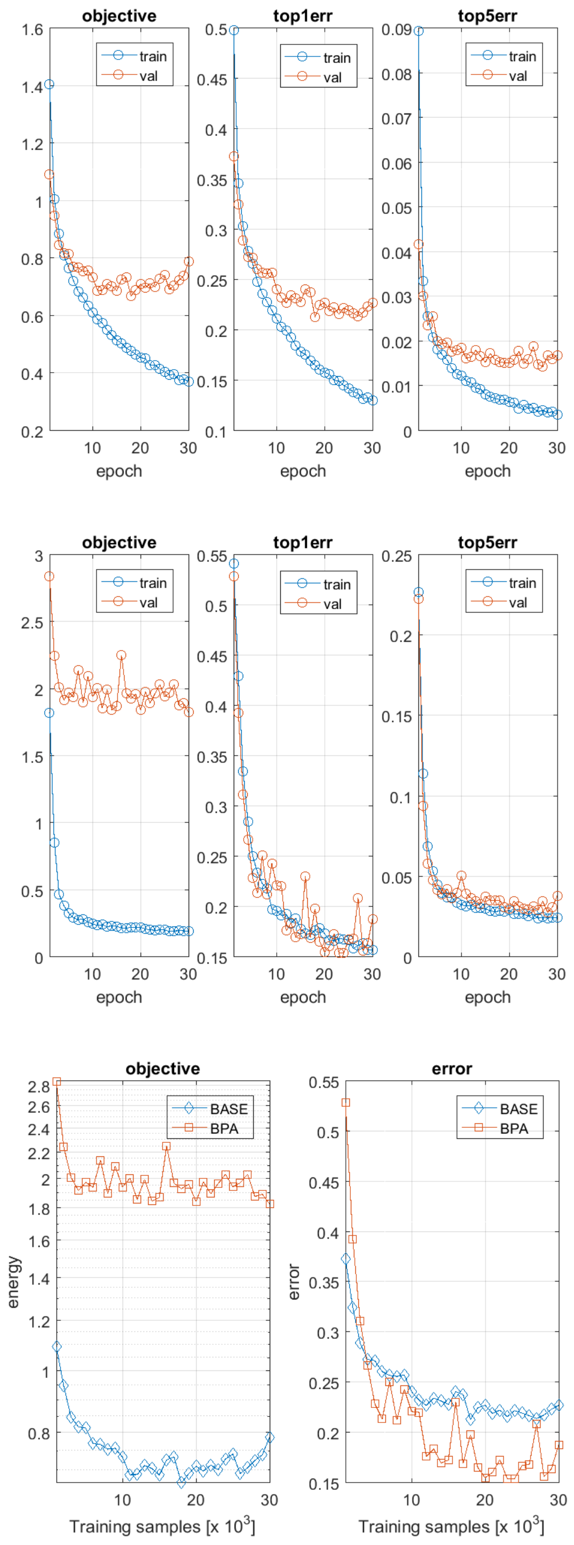

Figure 9.4: Unified backpropagation for CIFAR-10 dataset. Compared to the Softmax baseline (chart 1), the unification scheme (chart 2) generates larger objective values. Due to a better fit to the data distribution, validation error follows training error, quite closely and outperforms the baseline of Softmax. A higher level of validation energy (chart3) generates better performance, when the unified backpropagation is employed. With respect to MNIST dataset, some high fluctuation spots appear in the validation errors, but the overall trend shows a reasonably smooth decay rate.

9.4 Conclusion

In this chapter, a unified backpropagation scheme was proposed to update the filters in a multi-objective convolutional network. This was in contrast with standard practices for a CNN that generally employed multiple objective functions (SVM, Softmax, LDA), trained them independently and then, assembled together, based on their performances. Here, BPA merged the outputs of different objectives and backpropagated the error to update the filters from all the convolutional layers. The experiments confirmed that the proposed scheme outperformed other conventional multi-objective regimes.

Chapter 10

Conclusion

Visual scene understanding has always been a matter of interest to computer vision community. The emergence of deep neural networks, introduces automatic feature learning as a powerful approach to replace the feature hand-crafting, to address different tasks. Deep learning lets train the models with huge number of parameters and solve high-dimensional, non-convex optimization problems. One of the well-known statistical methods to tackle spaces of high dimensions, is discriminant analysis that imposes better class discrepancy. This thesis proposed Deep Fisher Discriminant Learning to link Fisher discrimination and deep learning. It targeted semantic segmentation, texture classification and object recognition as important challenges in visual scene understanding. The theoretical justifications, supported by experimental results, confirmed the advantage of the proposed algorithms to improve the performance over various standard benchmarks in the literature.

10.1 Summary

Following the introduction (Chapter 1) and literature review (Chapter 2), Chapter 3 introduced Fisher discriminant analysis (dimension reduction) and then, reformulated this to produce a supervised projection into a high-dimensional space, spanned by the number of classes in the dataset under study (dimension expansion).

Part I (Chapter 4) on semantic segmentation, proposed a supervised colour transformation based on Fisher discrimination. The process began with a supervised projection from primary colour space into a new space, spanned by the number of classes, and continued by backprojection into an intermediate colour space. The former, imposed distinction across classes by minimizing the ratio of inter-intra class scatterings, while the latter, exposed the distances in the new colour space by a subspace mapping. This space was scaled-translated to form the target colour space. The experimental results showed advantage of the supervised colour transformation, over other subspace mappings and colour spaces, to improve global and average precisions of pixelwise semantic segmentation.

Part II (Chapters 5, 6) on texture classification, introduced some new deep neural architectures to train banks of texture filters. The supremacy of deep convolutional neural networks for automatic feature learning, led to the introduction of Fisher convolutional neural network and Fisher convolutional autoencoder network. The experiments on several texture datasets and deep descriptors, confirmed the advantage of these deep architectures to produce considerable improvements over the recently published benchmarks. The Fisher convolutional neural network was found to be computationally efficient, due to the learning of filter parameters instead of whole texture filters, expanding into the large number of filters for better generalization, and moving towards deeper layers for quality features. The Fisher convolutional autoencoder network was able to train both of the parametric and separable texture filters. This employed Fisher discrimination to impose higher distinction among texture classes in the datasets at hand. It resulted in arranging along independent stacks, so as, to train each texture filter in different depths, based on the complexity of patterns under study, and the ability of each filter to extract high-order convolutional features.

Part III (Chapters 7, 8, 9) on object recognition, discussed some novel contributions toward pooling, transfer learning, and multi-objective learning in deep neural networks. They were all inspired by Fisher discrimination and basic probability assignment of the evidence theory. First, a new pooling scheme, called multipartite pooling, projected the activations to a new high-dimensional space and then, scored them by an accumulative bipartite ranking approach. These scores could be used to pick the highly informative activation instances in the pooling layers of any deep convolutional architectures.

Second, a novel distributed backpropagation strategy for transfer learning, tackled the optimization complexity of highly non-convex objective functions. This also resolved the problem of imbalanced data distributions of the primary and target domains, across several deep single-filter learners. Third, a unified backpropagation scheme, linked the gradients of multiple loss functions to update the parameters of a multi-objective network, at the time of backpropagation. This avoided biases in imbalanced data distributions, and improved the classification performance of single-multi objective learning for deep convolutional networks.

10.2 Future Work

To further extend the ideas presented in this thesis, the following lines of research should be considered for future work:

- In addition to the Fisher convolutional and Fisher autoencoder networks, Fisher discrimination may introduce a novel set of deep architectures, that optimize discriminating objectives, rather than common error functions. Since discriminant analysis is a distributed algorithm in nature, high-performance parallel deep learning could possibly be achieved in these paradigms.
- Multipartite ranking, transfer learning, and multi-objective learning by Fisher discrimination, may have applications in other areas of machine learning, such as, natural language processing and machine translation.
- Scene understanding is one of the hardest challenges in computer vision. This thesis showed several successful practices of novel deep neural architectures, that addressed some specific tasks, such as, segmentation, classification and recognition. This work has the potential to improve the performance of other visual processing applications, such as, in biometrics, surveillance and compression.

Bibliography

ABE, N.; ZADROZNY, B.; AND LANGFORD, J., 2004. An iterative method for multi-class cost-sensitive learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 3–11. ACM. (cited on page 110)

ADRAGNI, K. P. AND COOK, R. D., 2009. Sufficient dimension reduction and prediction in regression. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 367, 1906 (2009), 4385–4405. (cited on page 8)

ANDREW, G.; ARORA, R.; BILMES, J. A.; AND LIVESCU, K., 2013. Deep canonical correlation analysis. In *ICML (3)*, 1247–1255. (cited on page 107)

ANGIULLI, F., 2007. Fast nearest neighbor condensation for large data sets classification. *IEEE Transactions on Knowledge and Data Engineering*, 19, 11 (2007), 1450–1464. (cited on page 78)

BADRI, H.; YAHIA, H.; AND DAOUDI, K., 2014. Fast and accurate texture recognition with multilayer convolution and multifractal analysis. In *Computer Vision–ECCV 2014*, 505–519. Springer. (cited on page 43)

BAUDAT, G. AND ANOUAR, F., 2000. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12, 10 (2000), 2385–2404. (cited on page 17)

BECK, A. AND TEBOULLE, M., 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2, 1 (2009), 183–202. (cited on pages 48 and 60)

BENGIO, Y.; LAMBLIN, P.; POPOVICI, D.; LAROCHELLE, H.; ET AL., 2007. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19 (2007), 153. (cited on page 10)

BENGIO, Y.; THIBODEAU-LAUFER, E.; ALAIN, G.; AND YOSINSKI, J., 2013. Deep generative stochastic networks trainable by backprop. *arXiv preprint arXiv:1306.1091*, (2013). (cited on page 55)

BENGIO, Y. ET AL., 2012. Deep learning of representations for unsupervised and transfer learning. *ICML Unsupervised and Transfer Learning*, 27 (2012), 17–36. (cited on pages 97 and 104)

BICKEL, P. J. AND LEVINA, E., 2004. Some theory for fisher's linear discriminant function,'naive bayes', and some alternatives when there are many more variables than observations. *Bernoulli*, (2004), 989–1010. (cited on page 30)

BISHOP, C. M., 2006. *Pattern recognition and machine learning*. springer. (cited on pages 17, 31, 33, 47, 48, and 82)

BOHN, L. L. AND WOLFE, D. A., 1994. The effect of imperfect judgment rankings on properties of procedures based on the ranked-set samples analog of the mann-whitney-wilcoxon statistic. *Journal of the American Statistical Association*, 89, 425 (1994), 168–176. (cited on page 80)

BORG, I. AND GROENEN, P. J., 2005. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media. (cited on page 8)

BOUREAU, Y.-L.; PONCE, J.; AND LECUN, Y., 2010. A theoretical analysis of feature pooling in visual recognition. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 111–118. (cited on page 77)

BOURLARD, H. AND KAMP, Y., 1988. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59, 4-5 (1988), 291–294. (cited on page 11)

BREFELD, U. AND SCHEFFER, T., 2005. Auc maximizing support vector learning. In *Proceedings of the ICML 2005 workshop on ROC Analysis in Machine Learning*. Citeseer. (cited on page 78)

BROOMHEAD, D. S. AND LOWE, D., 1988. Radial basis functions, multi-variable functional interpolation and adaptive networks. Technical report, DTIC Document. (cited on page 10)

BURGES, C.; SHAKED, T.; RENSHAW, E.; LAZIER, A.; DEEDS, M.; HAMILTON, N.; AND HULLENDER, G., 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, 89–96. ACM. (cited on page 78)

CAMBRIA, E.; HUANG, G.-B.; KASUN, L. L. C.; ZHOU, H.; VONG, C. M.; LIN, J.; YIN, J.; CAI, Z.; LIU, Q.; LI, K.; ET AL., 2013. Extreme learning machines [trends & controversies]. *IEEE Intelligent Systems*, 28, 6 (2013), 30–59. (cited on page 12)

CHAN, T.-H.; JIA, K.; GAO, S.; LU, J.; ZENG, Z.; AND MA, Y., 2015. Pcanet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing*, 24, 12 (2015), 5017–5032. (cited on pages 107 and 109)

CHATFIELD, K.; SIMONYAN, K.; VEDALDI, A.; AND ZISSERMAN, A., 2014. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, (2014). (cited on page 60)

CHEN, L.-F.; LIAO, H.-Y. M.; KO, M.-T.; LIN, J.-C.; AND YU, G.-J., 2000. A new lda-based face recognition system which can solve the small sample size problem. *Pattern recognition*, 33, 10 (2000), 1713–1726. (cited on page 9)

CHONG, H. Y.; GORTLER, S. J.; AND ZICKLER, T., 2008. A perception-based color space for illumination-invariant image processing. *ACM Transactions on Graphics*, 27, 3 (2008), 61. (cited on page 29)

CHUNG, J.; GULCEHRE, C.; CHO, K.; AND BENGIO, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, (2014). (cited on page 12)

CIMPOI, M.; MAJI, S.; KOKKINOS, I.; MOHAMED, S.; ; AND VEDALDI, A., 2014a. Describing textures in the wild. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. (cited on page 49)

CIMPOI, M.; MAJI, S.; KOKKINOS, I.; MOHAMED, S.; AND VEDALDI, A., 2014b. Describing textures in the wild. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 3606–3613. IEEE. (cited on pages 50, 60, and 61)

CIMPOI, M.; MAJI, S.; KOKKINOS, I.; AND VEDALDI, A., 2015a. Deep filter banks for texture recognition, description, and segmentation. *arXiv preprint arXiv:1507.02620*, (2015). (cited on pages 43, 46, and 51)

CIMPOI, M.; MAJI, S.; AND VEDALDI, A., 2015b. Deep filter banks for texture recognition and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3828–3836. (cited on pages 43 and 61)

CLEVERT, D.-A.; MAYR, A.; UNTERTHINER, T.; AND HOCHREITER, S., 2015. Rectified factor networks. In *Advances in neural information processing systems*, 1855–1863. (cited on page 107)

COATES, A.; LEE, H.; AND NG, A. Y., 2010. An analysis of single-layer networks in unsupervised feature learning. *Ann Arbor*, 1001, 48109 (2010), 2. (cited on page 107)

COLEMAN, T. F. AND LI, Y., 1996. A reflective newton method for minimizing a quadratic function subject to bounds on some of the variables. *SIAM Journal on Optimization*, 6, 4 (1996), 1040–1058. (cited on pages 33, 49, 61, and 84)

COMBETTES, P. L. AND PESQUET, J.-C., 2011. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, 185–212. Springer. (cited on page 48)

CUNNINGHAM, J. P. AND GHAHRAMANI, Z., 2015. Linear dimensionality reduction: Survey, insights, and generalizations. *Journal of Machine Learning Research*, (2015). (cited on pages 7, 33, 57, and 83)

DASARATHY, B. V., 1994. Minimal consistent set (mcs) identification for optimal nearest neighbor decision systems design. *IEEE Transactions on Systems, Man, and Cybernetics*, 24, 3 (1994), 511–517. (cited on page 78)

DENG, X.; LIU, Q.; DENG, Y.; AND MAHADEVAN, S., 2016. An improved method to construct basic probability assignment based on the confusion matrix for classification problem. *Information Sciences*, 340 (2016), 250–261. (cited on pages 97, 100, and 110)

DESJARDINS, G. AND BENGIO, Y., 2008. Empirical evaluation of convolutional rbms for vision. *DIRO, Université de Montréal*, (2008), 1–13. (cited on page 13)

DONAHUE, J.; JIA, Y.; VINYALS, O.; HOFFMAN, J.; ZHANG, N.; TZENG, E.; AND DARRELL, T., 2013. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, (2013). (cited on pages 51, 60, and 61)

DORFER, M.; KELZ, R.; AND WIDMER, G., 2015. Deep linear discriminant analysis. *arXiv preprint arXiv:1511.04707*, (2015). (cited on pages 15, 107, 109, and 112)

ECKART, C. AND YOUNG, G., 1936. The approximation of one matrix by another of lower rank. *Psychometrika*, 1, 3 (1936), 211–218. (cited on page 7)

EIGEN, D.; ROLFE, J.; FERGUS, R.; AND LECUN, Y., 2013. Understanding deep architectures using a recursive convolutional network. *arXiv preprint arXiv:1312.1847*, (2013). (cited on page 13)

ELKAN, C., 2001. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, vol. 17, 973–978. LAWRENCE ERLBAUM ASSOCIATES LTD. (cited on pages 97 and 110)

ELMAN, J. L., 1990. Finding structure in time. *Cognitive science*, 14, 2 (1990), 179–211. (cited on page 11)

FAN, J. AND FAN, Y., 2008. High dimensional classification using features annealed independence rules. *The Annals of Statistics*, 36, 6 (2008), 2605. (cited on page 30)

FAWCETT, T., 2006. An introduction to roc analysis. *Pattern recognition letters*, 27, 8 (2006), 861–874. (cited on pages 80 and 110)

FISHER, R. A., 1936. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7, 2 (1936), 179–188. (cited on pages 8 and 9)

FREUND, Y.; IYER, R.; SCHAPIRE, R. E.; AND SINGER, Y., 2003. An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4, Nov (2003), 933–969. (cited on page 78)

FUKUNAGA, R., 1990. Statistical pattern recognition. (1990). (cited on pages 18 and 32)

FUKUSHIMA, K., 1988. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1, 2 (1988), 119–130. (cited on page 77)

GEUSEBROEK, J.-M.; VAN DEN BOOMGAARD, R.; SMEULDERS, A. W. M.; AND GEERTS, H., 2001. Color invariance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23, 12 (2001), 1338–1350. (cited on page 35)

GLOROT, X. AND BENGIO, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, 249–256. (cited on page 46)

GLOROT, X.; BORDES, A.; AND BENGIO, Y., 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 513–520. (cited on page 96)

GONG, M.; LIU, J.; LI, H.; CAI, Q.; AND SU, L., 2015. A multiobjective sparse feature learning model for deep neural networks. *IEEE transactions on neural networks and learning systems*, 26, 12 (2015), 3263–3277. (cited on page 108)

GOODFELLOW, I.; BENGIO, Y.; AND COURVILLE, A., 2016a. *Deep learning*. MIT Press. (cited on page 12)

GOODFELLOW, I.; BENGIO, Y.; AND COURVILLE, A., 2016b. Deep learning. http://goodfeli.github.io/dlbook/. Book in preparation for MIT Press. (cited on page 55)

GOODFELLOW, I.; POUGET-ABADIE, J.; MIRZA, M.; XU, B.; WARDE-FARLEY, D.; OZAIR, S.; COURVILLE, A.; AND BENGIO, Y., 2014. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2672–2680. (cited on page 12)

GOULD, S., 2012. Darwin: A framework for machine learning and computer vision research and development. *The Journal of Machine Learning Research*, 13, 1 (2012), 3533–3537. (cited on pages 31 and 35)

GOULD, S.; FULTON, R.; AND KOLLER, D., 2009. Decomposing a scene into geometric and semantically consistent regions. *IEEE 12th International Conference on Computer Vision (ICCV)*, (2009), 1–8. (cited on pages 31, 35, and 36)

GOWDA, K. C. AND KRISHNA, G., 1979. The condensed nearest neighbor rule using the concept of mutual nearest neighborhood. *IEEE Transactions on Information Theory*, 25, 4 (1979), 488–490. (cited on page 78)

GRAHAM, B., 2014. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, (2014). (cited on page 77)

GRAVES, A.; WAYNE, G.; AND DANIHELKA, I., 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*, (2014). (cited on page 12)

GUO, Y.; HASTIE, T.; AND TIBSHIRANI, R., 2007. Regularized linear discriminant analysis and its application in microarrays. *Biostatistics*, 8, 1 (2007), 86–100. (cited on page 9)

HE, H. AND GARCIA, E. A., 2009. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21, 9 (2009), 1263–1284. (cited on page 110)

HE, K.; ZHANG, X.; REN, S.; AND SUN, J., 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, (2015). (cited on page 12)

HE, X.; CAI, D.; AND MIN, W., 2005. Statistical and computational analysis of locality preserving projection. In *Proceedings of the 22nd international conference on Machine learning*, 281–288. ACM. (cited on page 8)

HERSHEY, J. R. AND OLSEN, P. A., 2007. Approximating the kullback leibler divergence between gaussian mixture models. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, vol. 4, IV–317. IEEE. (cited on page 80)

HILBORN, C., 1968. Dg lainiotis. *IEEE transactions on information theory*, (1968). (cited on page 78)

HINTON, G. E. AND SALAKHUTDINOV, R. R., 2006. Reducing the dimensionality of data with neural networks. *science*, 313, 5786 (2006), 504–507. (cited on page 13)

HINTON, G. E. AND SEJNOWSKI, T. J., 1986. Learning and releaming in boltzmann machines. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1 (1986), 282–317. (cited on page 10)

HOCHREITER, S. AND SCHMIDHUBER, J., 1997. Long short-term memory. *Neural computation*, 9, 8 (1997), 1735–1780. (cited on page 11)

HOPFIELD, J. J., 1982. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79, 8 (1982), 2554–2558. (cited on page 10)

HORN, R. A. AND JOHNSON, C. R., 2012. *Matrix analysis*. Cambridge university press. (cited on page 21)

HOTELLING, H., 1936. Relations between two sets of variates. *Biometrika*, 28, 3/4 (1936), 321–377. (cited on page 8)

HU, D.; BO, L.; AND REN, X., 2011a. Toward robust material recognition for everyday objects. In *BMVC*, vol. 13, 14. Citeseer. (cited on page 43)

HU, G.-N.; LIU, C.-C.; CHUANG, K.-W.; YU, S.-S.; AND TSUI, T.-S., 2011b. General regression neural network utilized for color transformation between images on rgb color space. *IEEE International Conference on Machine Learning and Cybernetics (ICMLC)*, 4 (2011), 1793–1799. (cited on page 29)

HUBEL, D. H. AND WIESEL, T. N., 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160, 1 (1962), 106–154. (cited on page 12)

HYVÄRINEN, A.; HOYER, P. O.; AND INKI, M., 2001. Topographic independent component analysis. *Neural computation*, 13, 7 (2001), 1527–1558. (cited on page 8)

JAEGER, H. AND HAAS, H., 2004. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, 304, 5667 (2004), 78–80. (cited on page 12)

JAIN, N.; THATTE, J.; BRACIALE, T.; LEY, K.; O'CONNELL, M.; AND LEE, J. K., 2003. Local-pooled-error test for identifying differentially expressed genes with a small number of replicated microarrays. *Bioinformatics*, 19, 15 (2003), 1945–1951. (cited on page 80)

JANKOWSKI, N. AND GROCHOWSKI, M., 2004. Comparison of instances seletion algorithms i. algorithms survey. In *International conference on artificial intelligence and soft computing*, 598–603. Springer. (cited on page 78)

JARRETT, K.; KAVUKCUOGLU, K.; LECUN, Y.; ET AL., 2009. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, 2146–2153. IEEE. (cited on pages 14 and 77)

JOHNSTONE, I. M. AND TITTERINGTON, D. M., 2009. Statistical challenges of high-dimensional data. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 367, 1906 (2009), 4237–4253. (cited on page 8)

KINGMA, D. P.; MOHAMED, S.; REZENDE, D. J.; AND WELLING, M., 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, 3581–3589. (cited on page 55)

KINGMA, D. P. AND WELLING, M., 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, (2013). (cited on page 11)

KOÇO, S. AND CAPPONI, C., 2013. On multi-class classification through the minimization of the confusion matrix norm. In *ACML*, 277–292. (cited on page 110)

KOHONEN, T., 1982. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43, 1 (1982), 59–69. (cited on page 10)

KRÄHENBÜHL, P. AND KOLTUN, V., 2012. Efficient inference in fully connected crfs with gaussian edge potentials. *arXiv preprint arXiv:1210.5644*, (2012). (cited on pages 31, 35, and 36)

KRIZHEVSKY, A. AND HINTON, G., 2009. Learning multiple layers of features from tiny images. (2009). (cited on pages 87 and 114)

KRIZHEVSKY, A. AND HINTON, G., 2010. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40 (2010). (cited on page 13)

KRIZHEVSKY, A.; SUTSKEVER, I.; AND HINTON, G. E., 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105. (cited on pages 12 and 43)

KULKARNI, T. D.; WHITNEY, W. F.; KOHLI, P.; AND TENENBAUM, J., 2015. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, 2539–2547. (cited on page 11)

LAZEBNIK, S.; SCHMID, C.; AND PONCE, J., 2005. A sparse texture representation using local affine regions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27, 8 (2005), 1265–1278. (cited on page 50)

LE CUN, B. B.; DENKER, J. S.; HENDERSON, D.; HOWARD, R. E.; HUBBARD, W.; AND JACKEL, L. D., 1990. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*. Citeseer. (cited on page 77)

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; AND HAFFNER, P., 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86, 11 (1998), 2278–2324. (cited on pages 11, 87, and 114)

LEE, S.-G. AND VU, Q.-P., 2011. Simultaneous solutions of sylvester equations and idempotent matrices separating the joint spectrum. *Linear Algebra and its Applications*, 435, 9 (2011), 2097–2109. (cited on pages 21 and 32)

LEUNG, T. AND MALIK, J., 2001. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International journal of computer vision*, 43, 1 (2001), 29–44. (cited on page 50)

LEYVA, E.; GONZÁLEZ, A.; AND PÉREZ, R., 2015. Three new instance selection methods based on local sets: A comparative study with several approaches from a bi-objective perspective. *Pattern Recognition*, 48, 4 (2015), 1523–1537. (cited on page 78)

LIU, L.; FIEGUTH, P.; CLAUSI, D.; AND KUANG, G., 2012. Sorted random projections for robust rotation-invariant texture classification. *Pattern Recognition*, 45, 6 (2012), 2405–2418. (cited on page 43)

LONG, M.; WANG, J.; AND JORDAN, M. I., 2016. Deep transfer learning with joint adaptation networks. *arXiv preprint arXiv:1605.06636*, (2016). (cited on page 95)

LOOG, M.; DUIN, R.; AND HAEB-UMBACH, R., 2001. Multiclass linear dimension reduction by weighted pairwise fisher criteria. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23, 7 (2001), 762–766. (cited on pages 30 and 35)

MAASS, W.; NATSCHLÄGER, T.; AND MARKRAM, H., 2002. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14, 11 (2002), 2531–2560. (cited on page 12)

MAKHZANI, A. AND FREY, B., 2013. K-sparse autoencoders. *arXiv preprint arXiv:1312.5663*, (2013). (cited on page 14)

MALLIKARJUNA, P.; FRITZ, M.; TARGHI, A. T.; HAYMAN, E.; CAPUTO, B.; AND EKLUNDH, J., 2006. The kth-tips and kth-tips2 databases. (cited on pages 50, 63, and 66)

MASCI, J.; MEIER, U.; CIREŞAN, D.; AND SCHMIDHUBER, J., 2011. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, 52–59. Springer. (cited on pages 14 and 55)

MATHIEU, M.; HENAFF, M.; AND LECUN, Y., 2013. Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851*, (2013). (cited on page 13)

MCLACHLAN, G., 2004. *Discriminant analysis and statistical pattern recognition*. John Wiley & Sons. (cited on page 30)

MEYER, G. W. AND GREENBERG, D. P., 1980. Perceptual color spaces for computer graphics. *ACM SIGGRAPH Computer Graphics*, 14, 3 (1980), 254–261. (cited on page 29)

MURRAY, N. AND PERRONNIN, F., 2014. Generalized max pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2473–2480. (cited on page 77)

NETZER, Y.; WANG, T.; COATES, A.; BISSACCO, A.; WU, B.; AND NG, A. Y., 2011. Reading digits in natural images with unsupervised feature learning. (2011). (cited on pages 88 and 114)

NUSSBAUM, M. AND SZKOŁA, A., 2009. The chernoff lower bound for symmetric quantum hypothesis testing. *The Annals of Statistics*, (2009), 1040–1057. (cited on page 80)

OLVERA-LÓPEZ, J. A.; CARRASCO-OCHOA, J. A.; AND MARTÍNEZ-TRINIDAD, J. F., 2010. A new fast prototype selection method based on clustering. *Pattern Analysis and Applications*, 13, 2 (2010), 131–141. (cited on page 78)

OQUAB, M.; BOTTOU, L.; LAPTEV, I.; AND SIVIC, J., 2014. Learning and transferring mid-level image representations using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, 1717–1724. IEEE. (cited on pages 43 and 96)

PAN, S. J. AND YANG, Q., 2010. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22, 10 (2010), 1345–1359. (cited on page 95)

PARK, H.; JEON, M.; AND ROSEN, J. B., 2003. Lower dimensional representation of text data based on centroids and least squares. *BIT Numerical mathematics*, 43, 2 (2003), 427–448. (cited on page 9)

PERRONNIN, F. AND LARLUS, D., 2015. Fisher vectors meet neural networks: A hybrid classification architecture. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3743–3752. (cited on page 16)

PETERSEN, K. B.; PEDERSEN, M. S.; ET AL., 2008. The matrix cookbook. *Technical University of Denmark*, 7 (2008), 15. (cited on pages 58 and 84)

PRASAD, M. M.; SUKUMAR, M.; AND RAMAKRISHNAN, A., 2010. Orthogonal lda in pca transformed subspace. In *Frontiers in Handwriting Recognition (ICFHR), 2010 International Conference on*, 172–175. IEEE. (cited on page 9)

RALAIVOLA, L., 2012. Confusion-based online learning and a passive-aggressive scheme. In *Advances in Neural Information Processing Systems*, 3284–3292. (cited on pages 97 and 110)

RANZATO, M.; POULTNEY, C.; CHOPRA, S.; AND LECUN, Y., 2006. Efficient learning of sparse representations with an energy-based model. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, 1137–1144. MIT Press. (cited on page 13)

RIFAI, S.; VINCENT, P.; MULLER, X.; GLOROT, X.; AND BENGIO, Y., 2011. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, 833–840. (cited on page 14)

RITTER, G.; WOODRUFF, H.; LOWRY, S.; AND ISENHOUR, T., 1975. An algorithm for a selective nearest neighbor decision rule. *IEEE Transactions on Information Theory*, 21, 6 (1975), 665–669. (cited on page 78)

ROSENBLATT, F., 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65, 6 (1958), 386. (cited on page 10)

RUSSAKOVSKY, O.; DENG, J.; SU, H.; KRAUSE, J.; SATHEESH, S.; MA, S.; HUANG, Z.; KARPATHY, A.; KHOSLA, A.; BERNSTEIN, M.; BERG, A. C.; AND FEI-FEI, L., 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115, 3 (2015), 211–252. doi:10.1007/s11263-015-0816-y. (cited on page 66)

SAINATH, T. N.; KINGSBURY, B.; MOHAMED, A.-R.; AND RAMABHADRAN, B., 2013. Learning filter banks within a deep neural network framework. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, 297–302. IEEE. (cited on page 13)

SAMMUT, C. AND WEBB, G. I., 2011. *Encyclopedia of machine learning*. Springer Science & Business Media. (cited on pages 49 and 100)

SÁNCHEZ, J. S.; BARANDELA, R.; MARQUÉS, A. I.; ALEJO, R.; AND BADENAS, J., 2003. Analysis of new techniques to obtain quality training sets. *Pattern Recognition Letters*, 24, 7 (2003), 1015–1022. (cited on page 78)

SCHMID, C., 2001. Constructing models for content-based image retrieval. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 2, II–39. IEEE. (cited on page 50)

SENTZ, K. AND FERSON, S., 2002. *Combination of evidence in Dempster-Shafer theory*, vol. 4015. Citeseer. (cited on pages 96, 100, and 101)

SERMANET, P.; EIGEN, D.; ZHANG, X.; MATHIEU, M.; FERGUS, R.; AND LECUN, Y., 2013. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, (2013). (cited on pages 98 and 104)

SHAHRIARI, A., 2016a. Learning deep filter banks in parallel for texture recognition. In *Image Processing (ICIP), 2016 IEEE International Conference on*, 1634–1638. IEEE. (cited on page 15)

SHAHRIARI, A., 2016b. Parametric learning of texture filters by stacked fisher autoencoders. In *Digital Image Computing: Techniques and Applications (DICTA), 2016 International Conference on*, 1–8. IEEE. (cited on page 16)

SHAHRIARI, A.; ALVAREZ, J. M.; AND ROBLES-KELLY, A., 2014. Class-driven color transformation for semantic labeling. In *Asian Conference on Computer Vision*, 436–451. Springer. (cited on page 9)

SHARAN, L.; ROSENHOLTZ, R.; AND ADELSON, E., 2009. Material perception: What can you see in a brief glance? *Journal of Vision*, 9, 8 (2009), 784–784. (cited on page 50)

SHOTTON, J.; WINN, J.; ROTHER, C.; AND CRIMINISI, A., 2009. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81, 1 (2009), 2–23. (cited on pages 31, 35, and 36)

SIMONYAN, K.; VEDALDI, A.; AND ZISSERMAN, A., 2013. Deep fisher networks for large-scale image classification. In *Advances in neural information processing systems*, 163–171. (cited on page 15)

SIMONYAN, K. AND ZISSERMAN, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, (2014). (cited on pages 43, 51, 60, and 61)

SIRONI, A.; TEKIN, B.; RIGAMONTI, R.; LEPETIT, V.; AND FUA, P., 2015. Learning separable filters. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37, 1 (2015), 94–106. (cited on page 66)

SMOLENSKY, P., 1986. Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document. (cited on page 10)

SPEARMAN, C., 1904. " general intelligence," objectively determined and measured. *The American Journal of Psychology*, 15, 2 (1904), 201–292. (cited on page 8)

STEELE, J. M., 2004. *The Cauchy-Schwarz master class: an introduction to the art of mathematical inequalities*. Cambridge University Press. (cited on page 24)

STRUTZ, T., 2012. Adaptive selection of colour transformations for reversible image compression. *IEEE 20th European Signal Processing Conference (EUSIPCO)*, (2012), 1204–1208. (cited on page 29)

STUHLSATZ, A.; LIPPEL, J.; AND ZIELKE, T., 2012. Feature extraction with deep neural networks by a generalized discriminant analysis. *IEEE transactions on neural networks and learning systems*, 23, 4 (2012), 596–608. (cited on pages 15 and 107)

SUN, M.; ZHANG, X.; ZHENG, T. F.; ET AL., 2016. Unseen noise estimation using separable deep auto encoder for speech enhancement. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24, 1 (2016), 93–104. (cited on page 14)

SWITZER, P. AND GREEN, A. A., 1984. Min/max autocorrelation factors for multivariate spatial imagery. *Computer science and statistics*, (1984), 13–16. (cited on page 8)

SYDOROV, V.; SAKURADA, M.; AND LAMPERT, C. H., 2014. Deep fisher kernels-end to end learning of the fisher kernel gmm parameters. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1402–1409. (cited on page 15)

TANG, Y., 2013. Deep learning using linear support vector machines. In *In ICML*. Citeseer. (cited on pages 107 and 112)

THEOBALD, C., 1975. An inequality for the trace of the product of two symmetric matrices. In *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 77, 265–267. Cambridge Univ Press. (cited on page 7)

TIBSHIRANI, R.; HASTIE, T.; NARASIMHAN, B.; AND CHU, G., 2002. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences*, 99, 10 (2002), 6567–6572. (cited on page 30)

TIBSHIRANI, R.; HASTIE, T.; NARASIMHAN, B.; AND CHU, G., 2003. Class prediction by nearest shrunken centroids, with applications to dna microarrays. *Statistical Science*, (2003), 104–117. (cited on page 9)

TIMOFTE, R. AND VAN GOOL, L. J., 2012. A training-free classification framework for textures, writers, and materials. In *BMVC*, vol. 13, 14. (cited on pages 43 and 50)

TOMEK, I., 1976. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on systems, Man, and Cybernetics*, , 6 (1976), 448–452. (cited on page 78)

UEMATSU, K. AND LEE, Y., 2015. Statistical optimality in multipartite ranking and ordinal regression. *IEEE transactions on pattern analysis and machine intelligence*, 37, 5 (2015), 1080–1094. (cited on page 78)

VALCARCEL MACUA, S.; BELANOVIC, P.; AND ZAZO, S., 2011. Distributed linear discriminant analysis. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, 3288–3291. IEEE. (cited on page 23)

VAN DE SANDE, K. E.; GEVERS, T.; AND SNOEK, C. G., 2010. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32, 9 (2010), 1582–1596. (cited on pages 29 and 35)

VARMA, M. AND ZISSERMAN, A., 2003. Texture classification: Are filter banks necessary? In *Computer vision and pattern recognition, 2003. Proceedings. 2003 IEEE computer society conference on*, vol. 2, II–691. IEEE. (cited on page 50)

VÁZQUEZ, F.; SÁNCHEZ, J. S.; AND PLA, F., 2005. A stochastic approach to wilsonâĂŹs editing algorithm. In *Iberian Conference on Pattern Recognition and Image Analysis*, 35–42. Springer. (cited on page 78)

VEDALDI, A. AND FULKERSON, B., 2008. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/. (cited on pages 87, 104, and 114)

VINCENT, P., 2011. A connection between score matching and denoising autoencoders. *Neural computation*, 23, 7 (2011), 1661–1674. (cited on page 14)

VINCENT, P.; LAROCHELLE, H.; LAJOIE, I.; BENGIO, Y.; AND MANZAGOL, P.-A., 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11, Dec (2010), 3371–3408. (cited on page 14)

VINYALS, O.; JIA, Y.; DENG, L.; AND DARRELL, T., 2012. Learning with recursive perceptual representations. In *Advances in Neural Information Processing Systems*, 2825–2833. (cited on page 107)

WAEGEMAN, W. AND DE BAETS, B., 2011. On the era ranking representability of pairwise bipartite ranking functions. *Artificial Intelligence*, 175, 7 (2011), 1223–1250. (cited on page 78)

WANG, S. AND YAO, X., 2012. Multiclass imbalance problems: Analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42, 4 (2012), 1119–1130. (cited on page 110)

WANG, X. AND SCHNEIDER, J., 2014. Flexible transfer learning under support and model shift. In *Advances in Neural Information Processing Systems*, 1898–1906. (cited on page 95)

WANG, X. AND TANG, X., 2004. Dual-space linear discriminant analysis for face recognition. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2 (2004), 564. (cited on page 17)

WILSON, D. L., 1972. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, , 3 (1972), 408–421. (cited on page 78)

WISKOTT, L., 2003. Slow feature analysis: A theoretical analysis of optimal free responses. *Neural Computation*, 15, 9 (2003), 2147–2177. (cited on page 8)

WU, L.; SHEN, C.; AND VAN DEN HENGEL, A., 2017. Deep linear discriminant analysis on fisher networks: A hybrid architecture for person re-identification. *Pattern Recognition*, 65 (2017), 238–250. (cited on page 16)

WYSZECKI, G. AND STILES, W. S., 1982. *Color science*, vol. 8. Wiley New York. (cited on pages 29 and 35)

XU, L.; KRZYZAK, A.; AND SUEN, C. Y., 1992. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE transactions on systems, man, and cybernetics*, 22, 3 (1992), 418–435. (cited on pages 97 and 110)

YE, J., 2005. Characterization of a family of algorithms for generalized discriminant analysis on undersampled problems. *Journal of Machine Learning Research*, 6, Apr (2005), 483–502. (cited on page 9)

YE, J.; CHEN, J.; JANARDAN, R.; AND KUMAR, S., 2008. Developmental stage annotation of drosophila gene expression pattern images via an entire solution path for lda. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2, 1 (2008), 4. (cited on page 9)

YOSINSKI, J.; CLUNE, J.; BENGIO, Y.; AND LIPSON, H., 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, 3320–3328. (cited on page 96)

ZEILER, M. D. AND FERGUS, R., 2013. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*, (2013). (cited on pages 77, 86, 87, and 91)

ZEILER, M. D.; KRISHNAN, D.; TAYLOR, G. W.; AND FERGUS, R., 2010. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2528–2535. IEEE. (cited on page 11)

ZHANG, K.; SCHÖLKOPF, B.; MUANDET, K.; AND WANG, Z., 2013. Domain adaptation under target and conditional shift. In *ICML (3)*, 819–827. (cited on page 95)

ZHAO, W.; CHELLAPPA, R.; AND PHILLIPS, P. J., 1999. *Subspace linear discriminant analysis for face recognition*. Citeseer. (cited on page 9)