FACULTY OF SCIENCE
UNIVERSITY OF COPENHAGEN

PhD Thesis

Deep Feature Learning and Cascaded Classifier for Large Scale Data

Segmenting Cartilage in Knee MRIs

Adhish Prasoon

adhish@diku.dk

Supervisor: Mads Nielsen

Co-supervisors: Francois Lauze, Marco Loog

Abstract

This thesis focuses on voxel/pixel classification based approaches for image segmentation. The main application is segmentation of articular cartilage in knee MRIs. The first major contribution of the thesis deals with large scale machine learning problems. Many medical imaging problems need huge amount of training data to cover sufficient biological variability. Learning methods scaling badly with number of training data points cannot be used in such scenarios. This may restrict the usage of many powerful classifiers having excellent generalization ability. We propose a cascaded classifier which allows usage of such classifiers in large scale problems. We demonstrate its application for segmenting tibial articular cartilage in knee MRI scans, with number of training voxels being more than 2 million. In the next phase of the study we apply the cascaded classifier to a similar but even more challenging problem of segmenting femoral cartilage. We discuss similarities and provide our solutions to the challenges. Our cascaded classifier for cartilage segmentation comprised of two stages of classification combining nearest neighbour classifier and support vector machine. We compared our method to a state-of-the-art method for cartilage segmentation using one stage nearest neighbour classifier. Our method achieved better results than the state-of-the-art method for tibial as well as femoral cartilage segmentation.

The next main contribution of the thesis deals with learning features autonomously from data rather than having a predefined feature set. We explore deep learning approach of convolutional neural network (CNN) for segmenting three dimensional medical images. We propose a novel system integrating three 2D CNNs, which have a one-to-one association with the xy, yz and zx planes of 3D image, respectively and this system is referred as triplanar convolutional neural network in the thesis. We applied the triplanar CNN for segmenting articular cartilage in knee MRI and compared its performance with the same state-of-the-art method which was used as a benchmark for cascaded classifier. Although our method used only 2D features at a single scale, it performs better than the state-of-the-art method using 3D multi-scale features. In the latter approach, the features and the

classifier have been carefully adapted to the problem at hand. That we were able to get better results by a deep learning architecture that autonomously learns the features from the images is the main insight of this study.

While training the convolutional neural networks for segmentation purposes, the commonly used cost function does not consider the labels of the neighbourhood pixels/voxels. We propose spatially contextualized convolutional neural network (SCCNN) which incorporates the labels of the neighbouring pixels/voxels while training the network. We demonstrate its application for the 2D problem of segmenting horses from the Weizmann horses database using 2D CNN and our 3D problem of segmenting tibial cartilage in knee MRIs using triplanar CNN. The proposed SCCNN improved the segmentation performance in both the cases. The good results obtained by SCCNN encourage to gain more insight into such frameworks.

Acknowledgements

I wish to express my sincere gratitude to my supervisor, Mads Nielsen, for his continuous support and guidance. My co-supervisor, François Lauze, has been extremely helpful through-out my PhD. I thank him for being always available for guiding me on my research and various related issues. I would also like to thank Marco Loog, with whom I had some valuable discussions. Christian Igel has been very supportive during my PhD. I am deeply grateful to him for all his crucial and valuable inputs, specially on machine learning. I express my gratitude to Kersten Petersen who introduced me to the field of deep learning and has ever been available for help. I am thankful to Erik Dam of Biomediq, for answering my queries on knee cartilage segmentation and also providing valuable feedback on my work. I am also thankful to Søren Olsen and Jon Sporring for being supportive and cooperative during my stint as a teaching assistant for their course. I would like to thank Dina Egholm, Susan Nasirumbi Ipsen and Camilla Jørgensen for helping me with matters related to administration, whenever needed. I am thankful to all the group members of the Image Group, who were very helpful and accommodating. I am specially thankful to Mattias Hansson, Chen Chen, Konstantin Chernoff, Sami Brandt, Akshay Pai, Jens Petersen, Kristoffer Smidt and Aasa Feragen for being such wonderful people to know and interact with. Finally, I would like to thank my parents, my uncle and aunt, my sisters and my wife for their love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Image segmentation is the process of partitionning an image into meaningful regions. These meaningful regions are useful for different types of analysis of the image. Practical applications of image segmentation include object detection, object recognition, face recognition, iris recognition, fingerprint recognition, automatic inspection, robotic guidance, video surveillance, content based image retrieval, satellite image classification etc.

1.1 Medical Image Segmentation

Image segmentation is a very important task for medical image analysis. Segmentation applications for medical images include finding important anatomical structures, locating tumors, measuring tissue volume etc. As mentioned in [2], it helps radiologists in visualization and study of the anatomical structures [3] , simulation of the biological processes [4], localization of pathologies [5], tracking the diseases [6], and evaluation of the necessity of radiotherapy or surgeries [7, 8]. Thus, a good segmentation is imperative for accurate quantitative analysis of the medical images and leads to better diagnosis and prognosis.

Segmentation is naturally performed in human brain. However, with the huge amount of images to process we can't rely on humans to perform the segmentations manually. Also, some fields need experts to perform the job, e.g. radiologists in medical imaging. Apart from being time taking and tedious for huge datasets,

manual segmentation is also prone to high inter and intra observer variability. After invention of image modalities which produce 3D/4D images (e.g. MRI, CT, ultrasound etc), the segmentation task has become even more difficult as the amount of data produced by these modalities are huge. Although the above mentioned facts are enough for justifying the need to perform automatic/semi-automatic segmentation, these segmentations should be good enough to be relied on. Specially for medical field where there is human life involved, we certainly don't want these methods to give false results and thus lead to improper/false diagnosis.

Pham et al [9] provided a nice survey of the most common segmentation methods in medical imaging. They divide medical image segmentation in several categories, and though relatively old (2000), their categorization is still accurate.

(1) Thresholding [10, 11, 12, 13].

(2) Region growing [14, 15, 16, 17]

(3) Classifiers [18, 19, 20]

(4) Clustering [21, 22]

(5) Markov random fields [23, 24]

(6) Artificial Neural Networks (ANN) [25, 26, 27, 28, 29]

(7) Deformable models [30, 31, 32, 33, 34, 35]

(8) Atlas-guided approach [36, 37, 38, 39]

(9) Other approaches (includes model fitting and watershed segmentation)[40, 41, 42, 43]

1.2 Osteoarthristis and Cartilage Segmentation in Knee MRI

Cartilage detoriaration causes Osteoarthristis, which is one of the most common diseases causing work disability. According to [44], approximately 250 million people suffer from osteoarthristis of knee(3.6 % of the world population). The number of hospitalizations due to osteoarthritis have increased from 322,000 in 1993 to 735,000 in 2006 in USA, as mentioned in [45]. Segmentation of articular cartilage in MRI scans is the method of choice for quantitative analysis of cartilage detoriaration. The quantitative anaylsis of cartilage can be used to develop efficient biomarkers for the osterarthristis diagnosis.

Our study involves low field MRI scans. Although the low field scanners have poorer resolution and image quality, yet they are more cost effective than high field scanners as they have lesser installation and maintainance cost. Also, patients find them more comfortable with almost no claustrophobic feeling present.

1.3 Machine Learning for Medical Image Segmentation

From the categorization given by [9], the methods based on classifiers and artificial neural networks can be classified as machine learning methods. Machine learning methods are very popular in medical imaging segmentation and are at the focus of this study.

In this study, we have applied our machine learning based methods for segmenting articular cartilage in knee MRIs. There are mainly three machine learning concepts used in this work, k nearest neighbours, support vector machines (SVMs) [46] and convolutional neural networks (CNNs) [47]. Support Vector Machines(SVMs) are one of the best classifiers and are one of the main machine learning concepts used in our work. We devote chapter 2 for the detailed discussion of SVMs. CNN is a deep learning method, which has recently been very successfully used for object recognition and segmentation. Chapter 5 discusses CNN in detail. The basic concept of kNN classifier is explained below.

k-Nearest Neighbours kNN is an instance based classifier in which we find k

nearest training data points of a query point. The neighbours are determined by the distances calculated in a feature space. Let $\mathbf{v}$ be a feature vector for the query data point. Out of the k training data points nearest to the query point, let n_c be the number of points belonging to a class c. The posterior probability of the query point belonging to class c is given by

$$p(c|\mathbf{v}) = \frac{n_c}{k}. \tag{1.1}$$

The posterior probability distribution is thresholded to assign the label. As kNN makes no assumption on data distribution, it is works well in most of the real life situations. While deciding upon the nearest neighbors the distances are calculated in the feature space which is also a metric space. In recent past it has been shown that a proper metric selection can be very important for the performance of the kNN [48, 49, 50, 51].

As kNN is an instance based classifier, we need the whole training data while making a prediction. The computational cost of naive kNN classifier is $O(nd)$, d being the number of features and n being the number of examples in the training data. [52, 53] proposed to speed-up kNN classification using KD-tree.

1.4 Organization of the Thesis

Rest of thesis is organized as follows.

Chapter 2 As mentioned earlier chapter 2 discusses Support Vector Machines in detail. This chapter follows closely the structure and content of the lectures notes of Christian Igel [1] on the topic and reproduces some of its content, with the explicit permission of Christian Igel.

Chapter 3 Many of the medical imaging tasks need huge amount of training data to cover sufficient biological variability. This restricts usage of some powerful classifiers which scale badly with number of training data points. Chapter 3 proposes a cascaded classifier which allows such classifiers to be used in scenarios

involving huge amount of training data and shows its application for segmenting tibial cartilage in knee MRIs. Chapter 3 is based on our article "Cascaded Classifier for Large-scale Data Applied to Automatic Segmentation of Articular Cartilage", published in the proceedings of SPIE Medical Imaging 2012: Image Processing Conference [54]. Apart from obvious formatting changes we have made some very minor changes in [54] to include it as a chapter in this thesis.

Chapter 4 In chapter 4 of this thesis, we apply our cascaded classifier to similar but even more challenging problem of segmenting femoral cartilage in knee MRIs. We discuss the similarities and explain our solutions to the challenges. Chapter 4 is a formatted and very minorly changed version of our article, "Femoral Cartilage Segmentation in Knee MRI Scans Using Two Stage Voxel Classification" by Adhish Prasoon, Christian Igel, Marco Loog, Francois Lauze, Erik Dam, and Mads Nielsen, published in the proceedings of 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC 2013) [55].

Chapter 5 Having appropriate features is vital for a classification task. Instead of using a predefined set of hand crafted features, in recent times, deep learning methods have been successfully used to autonomously learn features from the data. Convolutional Neural Network (CNN) is one such method and has been extensively used in this work. Chapter 5 introduces and discusses CNN in detail. The introductory part of CNN in this chapter is a formatted and enhanced version of "Deep Feature Learning for Knee Cartilage Segmentation Using a Triplanar Convolutional Neural Network" by Adhish Prasoon, Kersten Petersen, Christian Igel, Francois Lauze, Erik Dam, and Mads Nielsen, published in the proceedings of 16th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 2013) [56]. Backpropagation algorithm is used for calculating gradients while applying gradient based optimizations methods to train artificial neural networks . However, as per our knowledge it has not been well explained specifically for CNNs in the available literature. After introducing CNNs in the first part of the chapter, we discuss the backpropagation algorithm for CNNs in the second part.

Chapter 6 In chapter 6, we propose a novel method to segment 3D images using a system based on 2D CNNs which is computationally much less expensive to train and test than a 3D CNN and still performs better than a state-of-the-art method which uses a predefined set of multiscale features calculated using 3D kernels. We refer our system as *Triplanar Convolutional Neural Network*. While the introduction of CNN from [56] was already included in forming our chapter 5 , rest of [56] was formatted and very minorly changed to form chapter 6 of the thesis.

Chapter 7 In chapter 7, we propose and discuss novel *Spatially Contextualized Convolutional Neural Network (SCCNN)* which incorporates labels of neighborhood voxels/pixels while training the network. We demonstrate SCCNN's application to segment horses from Weizmann's horses' database [57] and its triplanar version's application to the problem of segmenting tibial cartilage in knee MRIs. We plan to submit an article based on this chapter in future.

Chapter 8 Finally chapter 8 concludes the thesis and discusses the potential future work.

Chapter 2

Support Vector Machines

2.1 Introduction

Support Vector Machines (SVMs) [46] are one of the best pattern recognition methods for binary classification. Idea building of SVM can be started with assuming an ideal scenario of two classes being linearly separable and the SVM trained to learn a hyperplane which maximizes the margin between the data points and the decision boundary. This means that the training of SVM is performed to find a hyperplane which has maximum distance from the data points nearest to it.

SVMs, which maximize the margin without allowing any margin violations are known as hard margin SVM. However, the overlap of classes is mostly unavoidable. Also, the hard-margin approach is very much sensitive to the noisy outliers. Even a single outlier has the capability to dramatically change the decision boundary. Practically, its better to allow few misclassifications on the training data in order to achieve better generalization. Thus, the concept of soft margin SVMs comes into picture. Soft margin SVMs allow the margins to be violated, however penalizing (not completely disallowing) such violations.

In many cases the input data is non-linear in nature and thus the input space is mapped into a higher dimensional space using a non-linear mapping. The algorithm now learns a hyperplane in this higher dimensional space to classify the patterns. We will see in further discussion, the SVM learning algorithm can be expressed using the input dot products. Calculating the features and their dot

products in the non linear space can be computationally very expensive. Using the kernel trick, the dot product in the non-linear space can be efficiently computed without even calculating the actual features in the non linear space. Kernel trick uses kernel functions which are real valued functions of two elements of the original input space. The case in which data is classified in the input space itself (without transforming the input space) corresponds to linear SVM, while the case in which input space is transformed into a non-linear space corresponds to non-linear SVM.

In the following discussion we start with discussing basic concept of linear optimal margin classifier and linear SVMs. Then we will discuss the kernel trick and the non-linear SVMs after which we move on to discuss the decomposition algorithms which are the most prominent algorithms for solving SVM optimization.

The discussion on SVM in this chapter follows closely the structure and content of the lectures notes of Christian Igel [1] on the topic and reproduces some of its content, with the explicit permission of Christian Igel.

2.2 Linear Optimal Margin Classifier

Binary classification is often based on a scalar discrimination function whose sign determines the class of the input pattern. Let $f : \mathcal{X} \to \mathbb{R}$ be a discrimination function and let $x \in \mathcal{X}$, $\mathcal{Y} = \{-1, 1\}$ be the input and the output of the hypothesis given by $h_f(x) = \text{sgn}(f(x))$, where $\text{sgn} : \mathbb{R} \to \{-1, 1\}$ is 1 if its argument is larger than zero and -1 otherwise.

Let $\mathcal{X} \subseteq \mathbb{R}^n$ and assume that the decision function f is affine. We can write the decision function as $f(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{w} \rangle + b$; determined (not uniquely) by parameters $(\boldsymbol{w}, b) \in \mathbb{R}^n \times \mathbb{R}$. The decision function partitions $\mathcal{X}$ into two half-spaces separated by a hyperplane $\ker(f) = \{\boldsymbol{x} \in \mathcal{X} \mid f(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{w} \rangle + b = 0\}$, defined in an $n - 1$ dimensional input subspace using $(\boldsymbol{w}, b) \in \mathbb{R}^n \times \mathbb{R}$. As all sets of parameters given by $(c\boldsymbol{w}, cb)$, $c \in \mathbb{R}^+$ lead to the same hyperplane and the same decision boundary, there is an *inherent degree of freedom* in the choice of parameters. The Euclidean distance of a point $\boldsymbol{x}_0$ from the hyperplane is given by the absolute value of $f(\boldsymbol{x}_0)/\|\boldsymbol{w}\|$. The side of hyperplane (half-space) to which the point belongs is determined by the sign of $f(\boldsymbol{x}_0)/\|\boldsymbol{w}\|$, see Fig. 2.1.

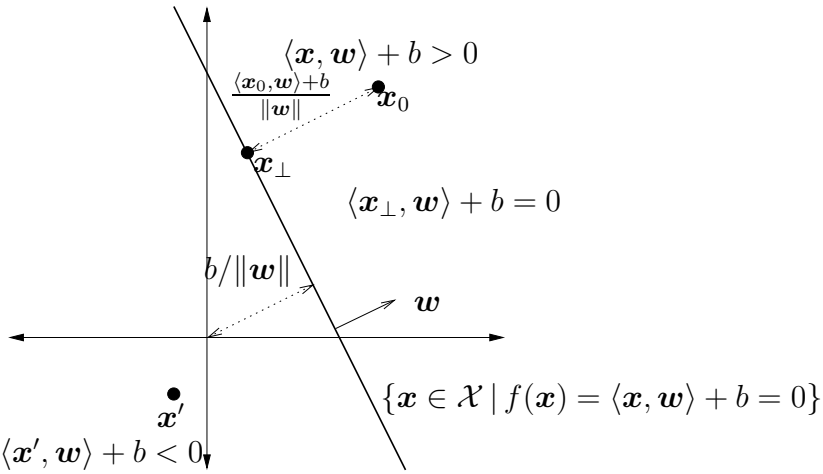


Figure 2.1: Linear classification according to a decision function $f(\boldsymbol{x}) = \langle \boldsymbol{x}, \boldsymbol{w} \rangle + b$ (reproduced from Christian Igel's lecture notes on the course "Statistical Machine Learning" [1], with permission from the author).

2.2.1 Margins

The *margin* of an input pattern $\boldsymbol{x}_i$ with respect to a hyperplane gives information about its distance from the hyperplane. It also tells whether the pattern lies to correct side of the hyperplane.

For an input pattern $(\boldsymbol{x}_i, y_i)$, the functional margin of the example with a hyperplane $(\boldsymbol{w}, b)$ is given as

$$\gamma_i = y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \ . \tag{2.1}$$

Moreover, the geometric margin of input pattern $(\boldsymbol{x}_i, y_i)$ with respect to the hyperplane $(\boldsymbol{w}, b)$ is defined as

$$\rho_i = y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b)/\|\boldsymbol{w}\| = \gamma_i/\|\boldsymbol{w}\| \ . \tag{2.2}$$

For the classification to be correct the margins should be positive. The absolute value of the geometric margin is the distance of the example from the hyperplane. A data set $S = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)\}$ is said to be linearly separable by $(\boldsymbol{w}, b)$ if for all $1 \leq i \leq \ell$, we have

$$y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) > 0 \tag{2.3}$$

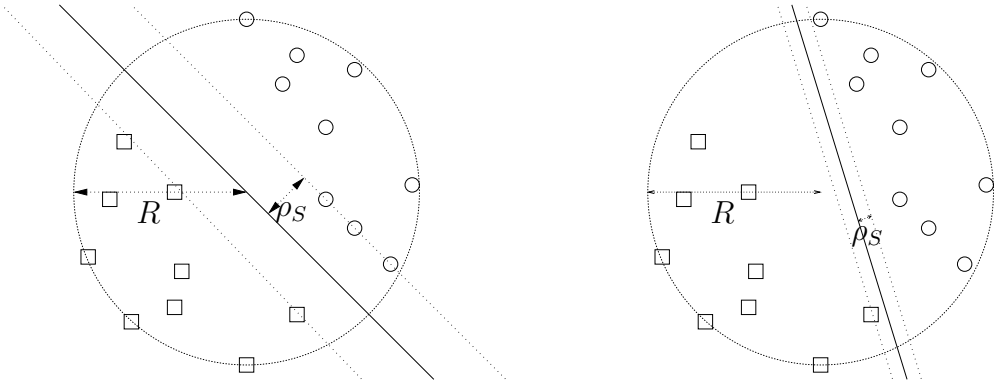


Figure 2.2: Linear classifiers, the hyperplane in the left figure classifies with maximum margin. The geometric margin with respect to the data set S is denoted by ρ_S, and R stands for the radius of the smallest ball containing S (reproduced from Christian Igel's lecture notes on the course "Statistical Machine Learning" [1], with permission from the author).

which means

$$y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq \gamma \tag{2.4}$$

for some $\gamma > 0$. Two hyperplanes that classify the patterns from a linearly separable set S correctly are shown in Figure 2.2 . The geometric and functional margin of a data set S with respect to a hyperplane $(\boldsymbol{w}, b)$ are given by $\rho_S = \min_{1 \leq i \leq \ell} \rho_i$ and $\gamma_S = min_{1 \leq i \leq \ell} \gamma_i$. The hyperplane which has maximum margin for a given dataset is called *maximum margin hyperplane*.

Before moving on to SVM optimization problem, we will discuss Lagranges duality in the next section, which will be helpful for further discussion on SVM.

2.2.2 Lagranges Duality

Let us define a constraint based optimization problem.

$$\left.\begin{array}{rll} \min\limits_{\boldsymbol{w}} f(\boldsymbol{w}) & & \boldsymbol{w} \in M \\ \text{subject to} \quad g_i(\boldsymbol{w}) \leq 0 & & i = 1, \ldots, k \\ h_j(\boldsymbol{w}) = 0 & & j = 1, \ldots, m \ , \end{array}\right\} \tag{2.5}$$

where functions f, g_i, h_j ($i = 1, \ldots, k$ and $j = 1, \ldots, m$) are defined on an open set $M \subset \mathbb{R}^n$.

This optimization problem is called a primal optimization problem. In order to solve this problem we need to construct the generalized lagrangian for the problem.

$$L(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{w}) + \sum_{i=1}^{m} \lambda_i h_i(\boldsymbol{w}) + \sum_{i=1}^{k} \mu_i g_i(\boldsymbol{w}) \ . \tag{2.6}$$

Here, $L(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ is the lagrangian and $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$ are the lagrangian multipliers. Now, let us consider the quantity

$$\beta_P = \max_{\boldsymbol{\lambda}, \boldsymbol{\mu}: \mu_i \geq 0} L(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{2.7}$$

If any of the constraints $h_i(\boldsymbol{w}) = 0$ is not respected, $\beta_P \to \infty$ by making the corresponding λ_i arbitrarily large i.e. $\lambda_i \to \infty$. Similarly if any of the constraints $g_i(\boldsymbol{w}) \leq 0$ is not respected, $\beta_P \to \infty$ by making the corresponding $\mu_i \to \infty$. If all the constraints are respected then we can verify that the maximum value attainable by $L(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ is $f(\boldsymbol{w})$. As we need to minimize $f(\boldsymbol{w})$ under above mentioned constraints, we can achieve that by minimizing the quantity β_P. Thus we can write our primal problem as

$$\min_{\boldsymbol{w}} \beta_P(\boldsymbol{w}) \tag{2.8}$$

which can be rewritten as

$$\min_{\boldsymbol{w}} \max_{\boldsymbol{\lambda}, \boldsymbol{\mu}: \mu_i \geq 0} L(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{2.9}$$

Now, let us define a quantity

$$\beta_D(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \min_{\boldsymbol{w}} L(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{2.10}$$

We can write the dual of the primal problem as

$$\max_{\boldsymbol{\lambda}, \boldsymbol{\mu}: \mu_i \geq 0} \beta_D(\boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{2.11}$$

We can rewrite the dual as

$$\max_{\boldsymbol{\lambda}, \boldsymbol{\mu}: \mu_i \geq 0} \min_{\boldsymbol{w}} L(\boldsymbol{w}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{2.12}$$

It can be shown that

$$\max_{\boldsymbol{\lambda},\boldsymbol{\mu}:\mu_i\geq 0} \min_{\boldsymbol{w}} L(\boldsymbol{w},\boldsymbol{\lambda},\boldsymbol{\mu}) \leq \min_{\boldsymbol{w}} \max_{\boldsymbol{\lambda},\boldsymbol{\mu}:\mu_i\geq 0} L(\boldsymbol{w},\boldsymbol{\lambda},\boldsymbol{\mu}) \tag{2.13}$$

Equality in the above equation can be found under some conditions, which are stated below.

Let f and the $g_i, \text{for } i = 1,\dots k$ are convex and the $h_j, \text{for } j = 1,\dots m$ are affine. Let us suppose that there exists some $\boldsymbol{w}$ so that $g_i(\boldsymbol{w}) < 0$ for all i. Assuming the above conditions hold true, there must exist $\boldsymbol{w}^*$ which provides solution of the primal problem and $\boldsymbol{\lambda}^*$, $\boldsymbol{\mu}^*$ provide the solution to the dual problem. Also

$$\max_{\boldsymbol{\lambda},\boldsymbol{\mu}:\mu_i\geq 0} \min_{\boldsymbol{w}} L(\boldsymbol{w},\boldsymbol{\lambda},\boldsymbol{\mu}) = \min_{\boldsymbol{w}} \max_{\boldsymbol{\lambda},\boldsymbol{\mu}:\mu_i\geq 0} L(\boldsymbol{w},\boldsymbol{\lambda},\boldsymbol{\mu}) = L(\boldsymbol{w}^*,\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*) \tag{2.14}$$

In addition to that, $\boldsymbol{w}^*, \boldsymbol{\lambda}^*$and $\boldsymbol{\mu}^*$ should satisfy the Karush-Kuhn-Tucker (KKT) conditions, given below:

$$\frac{\partial L(\boldsymbol{w}^*,\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*)}{\partial \boldsymbol{w}} = \mathbf{0} \ , \tag{2.15}$$

$$\frac{\partial L(\boldsymbol{w}^*,\boldsymbol{\lambda}^*,\boldsymbol{\mu}^*)}{\partial \boldsymbol{\lambda}} = \mathbf{0} \ , \tag{2.16}$$

$$\mu_i{}^* g_i(\boldsymbol{w}^*) = 0 \ \text{ for } i = 1,\dots,k \ , \tag{2.17}$$

$$g_i(\boldsymbol{w}^*) \leq 0 \ \text{ for } i = 1,\dots,k \ , \tag{2.18}$$

$$\mu_i{}^* \geq 0 \ \text{ for } i = 1,\dots,k \ . \tag{2.19}$$

where $\mu_i{}^*$ is the ith element of $\boldsymbol{\mu}^*$. Equation $\mu_i{}^* g_i(\boldsymbol{w}^*) = 0 \ \text{ for } i = 1,\dots,k$ is the the KKT complimentarity condition and has special importance in SVM theory.

2.2.3 Linear Hard Margin SVM

Assuming that there exists a hyperplane which can separate our data $S \in (\mathbb{R}^n \times \{-1,1\})^{\ell}$, a good choice for such a hyperplane should be the one which corresponds to maximum margin with respect to S. Training of linear hard margin

SVM involves solving an optimization problem of maximizing the margin under some constraints. The optimization problem can be written as following

$$\left.\begin{array}{ll} \max\limits_{\boldsymbol{w},b} & \rho = \gamma/\|\boldsymbol{w}\| \\ \text{subject to} & y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq \gamma \ , \ \ i = 1, \ldots, \ell \end{array}\right\} \tag{2.20}$$

γ being greater than zero, the constraint makes sure that the margins are positive for each pattern in S , making each of then classified correctly. The above formulation has an *inherent degree of freedom* as explained earlier. We can get rid of that by fixing $\gamma = 1$. Now we have a constraint optimization problem of maximizing $\rho = 1/\|\boldsymbol{w}\|$ which is same as minimizing $\frac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w} \rangle$ under same constraint. The optimization problem can now be written as

$$\left.\begin{array}{ll} \min\limits_{\boldsymbol{w},b} & \dfrac{1}{2}\langle \boldsymbol{w}, \boldsymbol{w} \rangle \\ \text{subject to} & y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) \geq 1 \ , \ \ i = 1, \ldots, \ell \ . \end{array}\right\} \tag{2.21}$$

The above optimization problem is in the primal form. The lagrangian in this case is given as

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{\ell} \alpha_i [y_i(\langle \boldsymbol{w}, x_i \rangle + b) - 1] \tag{2.22}$$

The primal problem in terms of its lagrangian can be written as

$$\min_{\boldsymbol{w},b} \max_{\boldsymbol{\alpha}:\alpha_i \geq 0} L(\boldsymbol{w}, b, \boldsymbol{\alpha})$$

The dual form of the above primal can be written as

$$\max_{\boldsymbol{\alpha}:\alpha_i \geq 0} \min_{\boldsymbol{w},b} L(\boldsymbol{w}, b, \boldsymbol{\alpha})$$

For the dual problem's solution to be same as the primal's, the following partial derivatives $\frac{\partial}{\partial \boldsymbol{w}} L(\boldsymbol{w}, b, \boldsymbol{\alpha})and\frac{\partial}{\partial b} L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ are set to zero (KKT condition)

$$\frac{\partial}{\partial \boldsymbol{w}} L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = 0 \qquad \frac{\partial}{\partial b} L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = 0 \tag{2.23}$$

As

$$\frac{\partial}{\partial \boldsymbol{w}} L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \boldsymbol{w} - \sum_{i=1}^{\ell} \alpha_i y_i \boldsymbol{x}_i \quad \text{and} \quad \frac{\partial}{\partial b} L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \sum_{i=1}^{\ell} \alpha_i y_i \ , \tag{2.24}$$

we have

$$\boldsymbol{w} = \sum_{i=1}^{\ell} \alpha_i y_i \boldsymbol{x}_i \tag{2.25}$$

and

$$\sum_{i=1}^{\ell} \alpha_i y_i = 0 \ . \tag{2.26}$$

Thus the dual optimization problem $\max_{\boldsymbol{\alpha}:\alpha_i \geq 0} \ \min_{\boldsymbol{w},b} L(\boldsymbol{w}, b, \boldsymbol{\alpha})$ can be written as

$$\left.\begin{array}{ll} \max\limits_{\boldsymbol{\alpha}} & \sum\limits_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum\limits_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \\ \text{subject to} & \sum\limits_{i=1}^{\ell} \alpha_i y_i = 0 \\ & \alpha_i \geq 0 \ , \ \ i = 1, \ldots, \ell \end{array}\right\} \tag{2.27}$$

Let $\boldsymbol{\alpha}^*$ be the solution of the dual problem defined above and ${\alpha^*}_i$ be its ith element. The solution of dual optimization problem is used to find the optimum $\boldsymbol{w}$ ($\boldsymbol{w}^*$) by the following equation (see equation (2.25)).

$$\boldsymbol{w}^* = \sum_{i=1}^{\ell} {\alpha^*}_i y_i \boldsymbol{x}_i \tag{2.28}$$

Once we find the $\boldsymbol{w}^*$, we can use that to find the optimum b (b^*)

$$b^* = -\frac{\max_{y_i=-1}(\langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle) + \min_{y_i=1}(\langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle)}{2} \ , \tag{2.29}$$

As we know the optimum hyperplane parameters, decision function can now be defined using $f(\boldsymbol{x}) = \langle \boldsymbol{w}^*, \boldsymbol{x} \rangle + b^*$. The KKT complimentarity is this case is

$${\alpha^*}_i [y_i(\langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle + b^*) - 1] = 0 \tag{2.30}$$

As we know that $y_i(\langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle + b^*) - 1 \geq 0$ for $i = 1 \ldots, \ell$, we can say that ${\alpha^*}_i = 0$ if $y_i(\langle \boldsymbol{w}^*, \boldsymbol{x}_i \rangle + b^*) - 1 > 0$. The points for which ${\alpha^*}_i > 0$ are the only points we need to define our decision boundary. These points are known as *support vectors*. Usually these points are much less in number than the total training data points.

While making the prediction for a new input $\boldsymbol{x}_i$, we calculate $(\langle \boldsymbol{w}^* \boldsymbol{x}_i \rangle + b)$ and predict $y_i = 1$ if this quantity is greater than 0 and $y_i = -1$ otherwise. The quantity $(\langle \boldsymbol{w}^* \boldsymbol{x}_i \rangle + b)$ can also be written as $(\langle \sum_{j=1}^{\ell} {\alpha^*}_j y_j \boldsymbol{x}_j \boldsymbol{x}_i \rangle + b)$ (see equation (2.30)). As we can see that to make a prediction we only need to perform its dot product with the support vectors, i.e having ${\alpha^*}_j > 0$. As the number of support vectors are much fewer than the number of training data points, the number of dot products to be performed for prediction is also small, making the prediction faster.

2.2.4 Linear Soft Margin SVM

In most of the real life problems its impossible to separate the data completely. Moreover, practically we don't want to force the separation of the data. By forcing the separability, we are giving the noisy outlier the capability to change the decision boundary drastically, leading to the overfitting of our decision function according to the training data.

For real world problems, we would like to have a classifier which is allowed to make few mistakes on the training data, in order to achieve better generalization. This leads us to the idea of soft margin SVMs. To "soften" our SVM, we need to re-formulate our problem for hard margin classifier in a way that it becomes less sensitive to the outliers. We do that by allowing functional margin to be less than one for training data points, but at the same time penalizing such margin violations.

We reformulate primal form of the linear hard-margin SVM problem to define the primal form of the soft-margin SVM as

$$\left.\begin{array}{rl} \min\limits_{\boldsymbol{\xi},\boldsymbol{w},b} & \dfrac{1}{2}\langle \boldsymbol{w},\boldsymbol{w}\rangle + C\sum\limits_{i=1}^{\ell}\xi_i \\ \text{subject to} & y_i(\langle \boldsymbol{w},\boldsymbol{x}_i\rangle + b) \geq 1-\xi_i \ , \ \ i=1,\ldots,\ell \\ & \xi_i \geq 0 \ , \ \ i=1,\ldots,\ell \end{array}\right\} \tag{2.31}$$

The above problem is a *linear soft-margin 1-norm SVM* problem. C is the regularization parameter. For training example $(\boldsymbol{x}_i, y_i)$ and hyperplane $(\boldsymbol{w}, b)$, variables ξ are the margin variables which are defined as below

$$\xi((\boldsymbol{x}_i, y_i), (\boldsymbol{w}, b), \gamma) = \xi_i = \max(0, 1 - y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b)) \ . \tag{2.32}$$

Thus for the data point $\boldsymbol{x}_i$ having functional margin greater than one, ξ is zero. If ξ is less than one, but greater than zero, then the data point violates the functional margin, but it is still on the correct side of the boundary. Finally the point is on the wrong side of the hyperplane if ξ is greater than one.

Similarly, a *linear soft-margin 2-norm SVM* problem can also be formulated as

$$\left.\begin{array}{rl} \min\limits_{\boldsymbol{\xi},\boldsymbol{w},b} & \dfrac{1}{2}\langle \boldsymbol{w},\boldsymbol{w}\rangle + \dfrac{C}{2}\sum\limits_{i=1}^{\ell}\xi_i^2 \\ \text{subject to} & y_i(\langle \boldsymbol{w},\boldsymbol{x}_i\rangle + b) \geq 1-\xi_i \ , \ \ i=1,\ldots,\ell \end{array}\right\} \tag{2.33}$$

Linear 1-norm Soft Margin SVM. The Lagrangian in this case can be written as

$$L(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{\ell}\xi_i - \sum_{i=1}^{\ell}\alpha_i[y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i\rangle + b) - 1 + \xi_i] - \sum_{i=1}^{\ell}\beta_i\xi_i \tag{2.34}$$

with constraints $\alpha_i, \beta_i \geq 0, i = 1, \ldots, \ell$. The KKT conditions are

$$\frac{\partial L}{\partial \boldsymbol{w}}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \boldsymbol{w} - \sum_{i=1}^{\ell} \alpha_i y_i x_i = 0 \ , \tag{2.35}$$

$$\frac{\partial L}{\partial \xi_i}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = C - \alpha_i - \beta_i = 0 \ \ i = 1, \ldots, \ell \ , \tag{2.36}$$

$$\frac{\partial L}{\partial b}(\boldsymbol{w}, b, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{i=1}^{\ell} \alpha_i y_i = 0 \ , \tag{2.37}$$

KKT complementarity conditions are given as below

$$\alpha_i [y_i(\langle \boldsymbol{w}, \boldsymbol{x}_i \rangle + b) - 1 + \xi_i] = 0 \ , \tag{2.38}$$

$$\beta_i \xi_i = 0 \ , \tag{2.39}$$

$$\xi_i(\alpha_i - C) = 0 \quad (\text{using } \beta_i = C - \alpha_i \text{ and } \beta_i \xi_i = 0) \tag{2.40}$$

for all $i = 1, \ldots, \ell$.

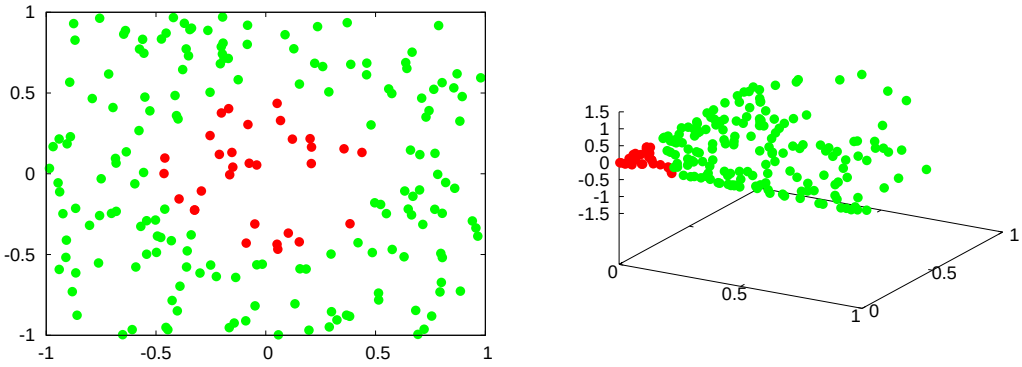
It can be observed from the complementarity conditions that the regularization parameter C imposes box constraints on the dual variables α_i.

Specifically, we get $0 \leq \alpha_i \leq C$ for all i from $\beta_i = C - \alpha_i$. $\xi_i(\alpha_i - C) = 0$ implies that if x_i violates the margin then the box constraint is active, $\alpha_i = C$.

In summary, the dual optimization problem can be written as

$$\left.\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \\ \text{subject to} \quad & \sum_{i=1}^{\ell} \alpha_i y_i = 0 \\ & C \geq \alpha_i \geq 0 \ , \ \ i = 1, \ldots, \ell \end{aligned} \right\} \tag{2.41}$$

We can also formulate the problem as *linear soft-margin 2-norm SVM* problem given as given in equation (2.33). We will not discuss its dual form and other details in this thesis. It can be formulated by constructing the lagrangian in a similar way as *1-norm soft margin SVM*.



(x_1, x_2) $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$

Figure 2.3: Example of an embedding into a feature space turning linearly non-separable to separable data. The colors indicate different class labels. The feature map $\Phi : \mathbb{R}^2 \to \mathbb{R}^3$ changes the representation of input patterns (x_1, x_2) to $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$ (reproduced from Christian Igel's lecture notes on the course "Statistical Machine Learning" [1], with permission from the author).

2.3 Non-linear SVMs and Kernels

As mentioned earlier, forcing the data separation is not a good idea and by using soft margin SVMs we get better generalization ability. However, its always desirable to have higher likelihood of data separability. As in many cases the input data is non-linear in nature, by non-linearly mapping the input space into a higher dimensional space, we can increase the likelihood of the data to be separable. Let $\mathcal{X}$, $\mathcal{F}$ be the original input space and the new space, and Φ: $\mathcal{X} \to \mathcal{F}$ be the non-linear *mapping*.

For example, we consider a polynomial classifier. Let the mapping $\Phi : \mathbb{R}^2 \to \mathbb{R}^3$ with $\Phi((x_1, x_2)) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$ be used to transform the input space. Figure 2.3 illustrates an example where the data that is not linearly separable in the original input space $\mathbb{R}^2$, but it is made linearly separable by $\Phi((x_1, x_2)$ in the new space $\mathbb{R}^3$.

Increasing the dimensionality can make the input data linearly separable, but on the other hand increases the computational costs. Fortunately many machine

learning algorithm require to compute only the dot product $\langle \Phi(x), \Phi(x') \rangle$ for $x, x' \in \mathcal{X}$ in the feature space. Therefore, the aim is to efficiently compute the dot product by a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle \tag{2.42}$$

for all $x, x' \in \mathcal{X}$, without actually perform mapping on the input patterns.

For a mapping to exist such that $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$ for all $x, x' \in \mathcal{X}$, a function has to be a valid kernel. Validity of a kernel is given by the Mercer's theorem, which is mentioned below.

$\boldsymbol{K}$ is defined as a kernel (Gram) matrix of k with respect to $x_1, \ldots, x_l$ if $\boldsymbol{K}$ has elements $K_{ij} = k(x_i, x_j)$. The rows and column of $\boldsymbol{K}$ are equal to the number of data examples.

2.3.1 Mercer's Theorem

The necessary and sufficient conditions for a given $k : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ to be a valid Mercer kernel is a) k is continuous b) k is symmetric c) For a given set $x_1, ..., x_l, (l < \infty)$, the corresponding Gram matrix $\boldsymbol{K}$ is symmetric positive semi-definite.

2.3.2 Gaussian Kernels

Gaussian kernels are the most commonly used kernels, specially when we don't know much about the problem at hand. For $\mathcal{X} = \mathbb{R}^n$, general Gaussian kernels are given as

$$k(\boldsymbol{x}, \boldsymbol{z}) = e^{-(\boldsymbol{x}-\boldsymbol{z})^T \boldsymbol{M} (\boldsymbol{x}-\boldsymbol{z})} \quad , \tag{2.43}$$

where $\boldsymbol{M}$ is a positive definite matrix. The common choice for $\boldsymbol{M}$ is $\boldsymbol{M} = \frac{1}{2\sigma^2}\boldsymbol{I}$, where $\boldsymbol{I}$ is a unit matrix and $\sigma \in \mathbb{R}^+$ is scalar parameter. For a radial Gaussian kernel, given as

$$k(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(-\|\boldsymbol{x} - \boldsymbol{z}\|^2/(2\sigma^2)\right), \tag{2.44}$$

the image of each element of $\mathcal{X}$ has unit length, thus, $k(\boldsymbol{x}, \boldsymbol{x}) = 1$ for $\boldsymbol{x} \in \mathcal{X}$, Moreover, the images of the elements of $\mathcal{X}$ lie in the same orthant, thus,

$\cos(\angle(\Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}))) = \langle \Phi(\boldsymbol{x}), \Phi(\boldsymbol{z}) \rangle = k(\boldsymbol{x}, \boldsymbol{z}) > 0$. Also, the radial Gaussian kernel has full ranked Gram matrix.

2.3.3 Deriving Kernels from Kernels.

We can derive new valid kernels from other valid kernels, which is useful in designing a kernel and therefore a representation for a particular problem. Assuming $k_1, k_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $k_3 : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ be valid kernels, $a \in \mathbb{R}^+$, $f : \mathcal{X} \to \mathbb{R}$, and $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^m$. The functions given below are positive definite kernels:

(1) $k(x, z) = a k_1(x, z)$,

(2) $k(x, z) = k_1(x, z) + k_2(x, z)$,

(3) $k(x, z) = k_1(x, z) k_2(x, z)$,

(4) $k(x, z) = e^{k_1(x,z)}$,

(5) $k(x, z) = f(x) f(z)$,

(6) $k(x, z) = k_3(\boldsymbol{\phi}(x), \boldsymbol{\phi}(z))$,

(7) $k(x, z) = k_1(x, z) / \sqrt{k_1(x, x) k_1(z, z)}$.

$k(x, x) \neq 0$ is assumed for all $x \in \mathcal{X}$ in the last case.

2.3.4 Kernel Trick

By using kernels we can have efficient formulations to various non linear variants of an algorithm expressed with the help of dot products. Quoting Schölkopf and Smola [58], this is known as the *kernel trick*: "Given an algorithm formulated in terms of a positive definite kernel k, one can construct an alternative algorithm by replacing k by an alternative kernel" [58].

We will not go into further details on this topic in this thesis. More details can be found in [58].

2.3.5 Non-linear Hard Margin SVM

The SVM optimization problem can be written using the dot products only. Thus, we can use the kernel trick and (see section 2.3.4) the dot products can be efficiently calculated using a valid kernel function k. By replacing the dot products with the kernel function k, we can simply convert the linear hard margin SVM into a non-linear pattern recognition algorithm. For training data $S = \{(x_1, y_1), \ldots, (x_\ell, y_\ell)\}$ and kernel function k to be linearly separable in the new feature space induced by k, the optimization problem

$$\left.\begin{array}{rl} \max\limits_{\boldsymbol{\alpha}} & \displaystyle\sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ \text{subject to} & \displaystyle\sum_{i=1}^{\ell} \alpha_i y_i = 0 \ , \ \alpha_i \geq 0 \ , \ i = 1, \ldots, \ell \end{array}\right\} \tag{2.45}$$

having solutions solution $\boldsymbol{\alpha}^*$, b^*, will give a decision rule $\text{sgn}(f(x))$, where $f(x) = \sum_{i=1}^{\ell} y_i {\alpha^*}_i k(x_i, x) + b^*$.

2.3.6 Non-linear Soft Margin SVM

Similar to the hard margin formulation, we can convert the linear algorithm into non-linear methods by applying kernel trick.

Non-linear 1-norm soft margin SVM. The optimization problem for non-linear 1-norm soft margin SVM can be written as

$$\left.\begin{array}{rl} \max\limits_{\boldsymbol{\alpha}} & \displaystyle\sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \\ \text{subject to} & \displaystyle\sum_{i=1}^{\ell} \alpha_i y_i = 0 \ , \ C \geq \alpha_i \geq 0 \ , \ i = 1, \ldots, \ell \end{array}\right\} \tag{2.46}$$

The above formulation leads to the decision rule of $\text{sgn}(f(x))$ where $f(x) = \sum_{i=1}^{\ell} y_i {\alpha^*}_i k(x_i, x) + b^*$.

We can see that for $C \to \infty$, the soft margin algorithm will be actually a hard margin SVM algorithm. Similar to the *Non-linear 1-norm soft margin SVM* case we can have a *Non-linear 2-norm soft margin SVM* case. However, we will not discuss it in this thesis. The 1-norm regularization is usually given preference over the 2-norm regularization as it leads to sparser solutions [59].

We have seen that finding an optimal decision function for SVM requires the constraint convex optimization problem to be solved. In the next section we will discuss an algorithm which solves our problem in an efficient way.

2.4 Training Support Vector Machines

Different methods have been proposed to solve constraint convex optimization problems for training SVMs (see [60]). These optimization problems are solvable using standard methods from quadratic programming. However, the SVM quadratic programs have a special form, which enables the usage of specialized heuristics with better time and space complexity.

We focus on training 1-norm soft margin SVMs, because of their popularity. Also, we have used them in our work. Although there are variety of different approaches to SVM training, we will discuss only *decomposition algorithms* [61, 62] which are very popular and highly efficient.

2.4.1 Decomposition Algorithms

Decomposition algorithms are the most prominent algorithms for solving SVM optimization [61, 63, 62, 64, 65, 66, 67, 68, 69, 70]. They are called decomposition algorithms because they break the learning problem into smaller subproblems. They restrict the dual optimization to a subset B (working set) and solve it iteratively. Algorithm 2.1 makes the procedure clear.

Dual optimization problem for 1-norm soft margin SVM (restricted to a work-

Algorithm 2.1: Decomposition Algorithm (reproduced from Christian Igel's lecture notes on the course "Statistical Machine Learning" [1], with permission from the author).

$\boldsymbol{\alpha} \leftarrow$ feasible starting point;
repeat
  select working set B;
  solve quadratic program restricted to B resulting in $\hat{\boldsymbol{\alpha}}$;
  $\boldsymbol{\alpha} \leftarrow \hat{\boldsymbol{\alpha}}$;
until *stopping criterion is met* ;

ing set $B \subset \{1, \ldots, \ell\}$)is as following

$$\left.\begin{aligned} \max_{\hat{\boldsymbol{\alpha}}} \quad & \mathcal{D}(\hat{\boldsymbol{\alpha}}) = \sum_{i=1}^{\ell} \hat{\alpha}_i - \frac{1}{2} \sum_{i,j=1}^{\ell} \hat{\alpha}_i \hat{\alpha}_j y_i y_j k(x_i, x_j) \\ \text{subject to} \quad & \sum_{i=1}^{\ell} \hat{\alpha}_i y_i = 0 \\ & \forall i \in \{1, \ldots, \ell\} : 0 \leq \hat{\alpha}_i \leq C \\ & \forall i \notin B : \hat{\alpha}_i = \alpha_i \ . \end{aligned}\right\} \tag{2.47}$$

Figure 2.4 visualizes this optimization problem with working set $B = \{i, j\}$ comprising of two variables.

We will use shortcut notation $K_{ij} = k(x_i, x_j)$ for Gram matrix entries and

$$g_i = \frac{\partial \mathcal{D}(\boldsymbol{\alpha}^*)}{\partial \alpha_i} = 1 - y_i \sum_{j=1}^{\ell} y_j {\alpha_j}^* K_{ij} \tag{2.48}$$

for the gradients. Also, let the index sets be defined as

$$I_{\text{up}} = \{i \,|\, y_i \alpha_i < b_i\} \qquad (y_i \alpha_i \text{ may increase}) \tag{2.49}$$

$$I_{\text{down}} = \{i \,|\, y_i \alpha_i > a_i\} \qquad (y_i \alpha_i \text{ may decrease}) \tag{2.50}$$

where

$$[a_i, b_i] = \begin{cases} [0, C] & \text{if } y_i = +1 \\ [-C, 0] & \text{if } y_i = -1 \end{cases} . \tag{2.51}$$

A support vector x_i is *bounded* if $\alpha_i = C$, and *free* otherwise (and $i \in I_{\text{up}} \wedge i \in I_{\text{down}}$).

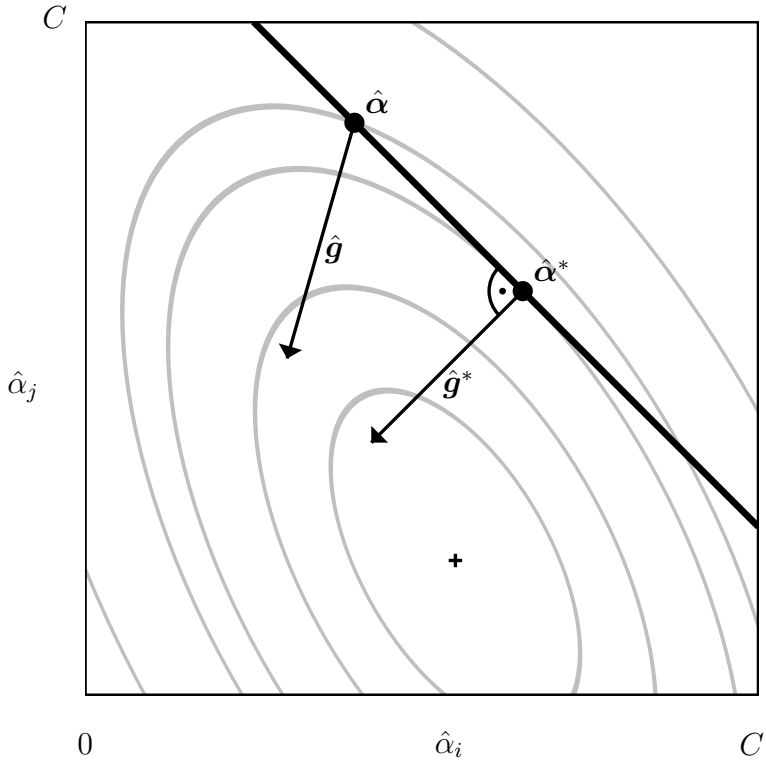


Figure 2.4: Two-dimensional subproblem for a 1-norm soft margin SVM. The restriction of the feasible region to the line segment in the box is due to the equality constraint and the box constraints. The points $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\alpha}}^*$ are feasible solutions of the subproblem, $\hat{\boldsymbol{\alpha}}^*$ is optimal. The corresponding gradients are denoted by $\hat{\boldsymbol{g}}$ and $\hat{\boldsymbol{g}}^*$ (reproduced from Christian Igel's lecture notes on the course "Statistical Machine Learning" [1], with permission from the author).

Optimality and Stopping Criterion A *stopping criterion* is required for algorithms which iteratively solve the optimization problem. In order to a find meaningful termination condition, a "quantitative" optimality criterion is found. $\boldsymbol{\alpha}^\epsilon \in \mathbb{R}^\ell$ for $\epsilon > 0$, $i \in I_{\text{up}}$ and $j \in I_{\text{down}}$ is defined as

$$\alpha_k^\epsilon = \alpha_k + \epsilon[\boldsymbol{u}_{ij}]_k = \alpha_k + \begin{cases} +\epsilon y_k & \text{if } k = i \\ -\epsilon y_k & \text{if } k = j \\ 0 & \text{otherwise} \end{cases}, \tag{2.52}$$

with $[\boldsymbol{u}_{ij}]_k$ being the kth component of the ℓ-dimensional vector $\boldsymbol{u}_{ij}$. $[\boldsymbol{u}_{ij}]_k$ is

0 except in its ith and jth components. The ith and jth components are equal to y_k and $-y_k$, respectively. Using the first order approximation we can write

$$\mathcal{D}(\boldsymbol{\alpha}^\epsilon) - \mathcal{D}(\boldsymbol{\alpha}^*) = \epsilon(y_i g_i^* - y_j g_j^*) + o(\epsilon) \ . \tag{2.53}$$

For $\boldsymbol{\alpha}^*$ to be optimal, the term $\mathcal{D}(\boldsymbol{\alpha}^\epsilon) - \mathcal{D}(\boldsymbol{\alpha}^*) = \epsilon(y_i g_i^* - y_j g_j^*) + o(\epsilon)$ has to be negative, which gives us the the necessary *optimality criterion*

$$\exists r \in \mathbb{R} : \max_{i \in I_{\text{up}}} y_i g_i^* \leq r \leq \min_{j \in I_{\text{down}}} y_j g_j^* \tag{2.54}$$

or

$$\exists r \in \mathbb{R} : \forall k : \begin{cases} \alpha_k^* = C & \text{if } g_k^* > y_k r \\ \alpha_k^* = 0 & \text{if } g_k^* < y_k r \end{cases} \ . \tag{2.55}$$

It can be shown that the optimality criterion is also sufficient. Let $\boldsymbol{\alpha}^*$ be a feasible solution of $\mathcal{D}$ and select

$$\boldsymbol{w}^* = \sum_{i=1}^{\ell} y_i \alpha_i^* k(x_i, \cdot) \ , \ \ b^* = r \ , \ \ \xi_i^* = \max\{0, g_i^* - y_i r\} \ . \tag{2.56}$$

Now the duality gap can be given as

$$\mathcal{P}(\boldsymbol{\xi}^*, \boldsymbol{w}^*, b^*) - \mathcal{D}(\boldsymbol{\alpha}^*) = C \sum_{i=1}^{\ell} \xi_i^* - \sum_{i=1}^{\ell} \alpha_i^* g_i^* = \sum_{i=1}^{\ell} (C\xi_i^* - \alpha_i^* g_i^*) \ , \tag{2.57}$$

where $\mathcal{P}(\boldsymbol{\xi}^*, \boldsymbol{w}^*, b^*)$ denotes the corresponding primal problem. As $C\xi_i^* - \alpha_i^* g_i^* = -y_i \alpha_i^* r$ we have

$$\mathcal{P}(\boldsymbol{\xi}^*, \boldsymbol{w}^*, b^*) - \mathcal{D}(\boldsymbol{\alpha}^*) = -r \sum_{i=1}^{\ell} y_i \alpha_i^* = 0 \ . \tag{2.58}$$

So, the duality gap is zero and $\boldsymbol{\alpha}^*$ is optimal. Moreover, setting $b^* = r$ is a nice way to determine b^*.

The optimality criterion can be computed by

$$\max_{i \in I_{\text{up}}} y_i g_i^* - \min_{j \in I_{\text{down}}} y_j g_j^* \leq 0 \ . \tag{2.59}$$

However, practically, this condition is loosened as given below

$$\max_{i \in I_{\text{up}}} y_i g_i - \min_{j \in I_{\text{down}}} y_j g_j \leq \epsilon \tag{2.60}$$

for $\epsilon > 0$. [69, 70] have shown that it is indeed a meaningful criterion with respect to accuracy of the solution to the primal problem. A good selection of ϵ can significantly speed-up SVM training without any significant loss of accuracy.

2.4.2 Recomputing Gradient and Stopping Criterion

As it is clear form the previous section gradient calculation plays an important role in decomposition algorithms. As $\boldsymbol{\alpha} = \mathbf{0}$ is always a feasible point with gradient at $\boldsymbol{\alpha} = \mathbf{0}$ evaluating to $\mathbf{1}$, it is a convenient starting point. The gradient vector of the full problem has to be adjusted after the solving the problem to the restricted set B, which can be done incrementally as following

$$\forall k \in \{1, \ldots, \ell\} : g_k \leftarrow g_k - y_k \sum_{i \in B} y_i (\hat{\alpha}_i - \alpha_i) K_{ik} \ . \tag{2.61}$$

2.4.3 Sequential Minimal Optimization

As we need to follow the equality constraint $\sum_{i=0}^{\ell} \alpha_i y_i = 0$, it is impossible to change just one variable and keep the rest fixed. Thus the minimum feasible working set size is two. In order to change some variable α_i by an amount of $\Delta\alpha_i$, we have to change another variable α_j too, by $-y_j y_i \Delta\alpha_i$. The decomposition algorithm which use working set size of two (minimal working set size) are known as SMO-type decomposition algorithm. SMO stands for *sequential minimal optimization* [62]. SMO is shown in Fig. 2.4.

Considering two variables has an advantage that the restricted two-dimensional subproblem is an analytically solvable problem. The working set $B = \{i, j\}$ comprises of $i \in I_{\text{up}}$ and $j \in I_{\text{down}}$. Let in an SMO step, $y_i \alpha_i$ increases, while $y_j \alpha_j$ decreases accordingly. Thus we can assume that $y_i g_i > y_j g_j$, which is taken care by the working set selection algorithms [67, 68]. The stopping condition defined in the previous section is satisfied if there is no pair with $i \in I_{\text{up}}$, $j \in I_{\text{down}}$, and $y_i g_i > y_j g_j$

Without loss of generality, if $i < j$, search direction of SMO in the subproblem is given by

$$(0, \ldots, y_i, 0, \ldots, 0, -y_j, 0, \ldots, 0) = \boldsymbol{u}_{ij} \ . \tag{2.62}$$

If we solve the subproblem in the search direction $\boldsymbol{u}_{ij}$ ignoring box constraints, it is same as maximizing

$$\mathcal{D}(\boldsymbol{\alpha} + \lambda \boldsymbol{u}_{ij}) - \mathcal{D}(\boldsymbol{\alpha}) = \lambda(y_i g_i - y_j g_j) - \frac{\lambda^2}{2}(K_{ii} + K_{jj} - 2K_{ij}) \tag{2.63}$$

with respect to λ. Newton step gives the optimal λ^*

$$\lambda^* = \frac{y_i g_i - y_j g_j}{K_{ii} + K_{jj} - 2K_{ij}} \tag{2.64}$$

λ^*. λ^* is then clipped to meet the box constraints

$$\lambda = \min \left\{ b_i - y_i \alpha_i, y_j \alpha_j - a_j, \frac{y_i g_i - y_j g_j}{K_{ii} + K_{jj} - 2K_{ij}} \right\} \tag{2.65}$$

The new coefficients are $\boldsymbol{\alpha} + \lambda \boldsymbol{u}_{ij}$. The SMO algorithm is shown in Algorithm 2.2.

Algorithm 2.2: Sequential minimal optimization (reproduced from Christian Igel's lecture notes on the course "Statistical Machine Learning" [1], with permission from the author).

$\boldsymbol{\alpha} \leftarrow \mathbf{0}$, $\boldsymbol{g} \leftarrow \mathbf{1}$;

repeat

select indices $i \in I_{\text{up}}$ and $j \in I_{\text{down}}$ with $y_i g_i > y_j g_j$;

$\lambda = \min \left\{ b_i - y_i \alpha_i, y_j \alpha_j - a_j, \frac{y_i g_i - y_j g_j}{K_{ii} + K_{jj} - 2K_{ij}} \right\}$;

$\forall k \in \{1, \ldots, \ell\} : g_k \leftarrow g_k - \lambda y_k K_{ik} + \lambda y_k K_{jk}$;

$\alpha_i \leftarrow \alpha_i + y_i \lambda$;

$\alpha_j \leftarrow \alpha_j - y_j \lambda$;

until $\max_{i \in I_{\text{up}}} y_i g_i - \min_{j \in I_{\text{down}}} y_j g_j \leq \epsilon$;

2.5 Training Time Scaling with Number of Patterns

[70] found the bounds for the time required to solve optimization problems of SVMs up to a certain accuracy while using decomposition algorithms. We mention two intuitive bounds given by [60]

- Assuming that an oracle tells us unbounded $F = \{x_i \,|\, 0 < \alpha_i < C\}$ and bounded support vectors, computing $\boldsymbol{\alpha}^*$ need just to solve an $|F|$-dimensional unconstrained optimization problem, which requires $O(|F|^3)$ computations.

- Computing the gradient from scratch is needed to check the optimality condition. It takes $O(\ell \cdot |\text{SV}|)$, where $|\text{SV}|$ denotes the number of support vectors, providing a strict lower bound.

The above mentioned are lower bounds. Actually, SVM training scale between quadratically and cubically with the training data population. Joachims et al. empirically showed this in [63]. Practically, the training time depends on the actual Gram matrix, the number of bounded support vectors (depends on the choice of C), and the stopping criterion. [70] showed that $O(1/\epsilon)$ iterations are sufficient to achieve an accuracy of ϵ. The dependence on the stopping criterion has special importance in this work, which will be evident in one of the following chapters.

2.6 SVM in our work

Intention behind using Support Vector Machines in this work was to exploit their excellent generalization ability in case of large-scale training data, which is a very common scenario in Medical Image Analysis. The need of large scale data comes because of the need to cover sufficient biological variability. Standard non-linear SVMs can't be used under such scenarios as they scale badly with the number of training data points. As an application, we took the voxel classification problem in knee MRIs and we aimed at improving the performance achieved by a state-of-the-art method proposed by [71]. As the training data used by [71] had more than 2 million voxels extracted from 25 scans and we wanted to exploit the whole of training data, the usage of standard non-linear SVM was practically not possible in this case as SVMs scale badly with number of training data points. [71] used kNN with feature selection. They started with a pool of 178 dimensional feature vector and reduced its dimensionality using feature selection. The training data had a huge class imbalance with background voxels outnumbering foreground voxels. We used their classifier by [71] as a screening stage and set the posterior threshold such that the false negatives are minimized. Now, just a fraction of voxels which are not classified as background are given to SVM for training. Although the kNN was fed

with selected features, as the dimensionality of feature space is not a problem with SVM, we decided to use the whole 178 dimensional feature set to given as input to the SVM. Hard margin SVMs are practically of no use because of the unavoidable overlap between the two classes. In our preliminary experiments, we used linear SVM but the results were clearly worse than [71]. For using non-linear SVM as our second stage, we had some options with respect to the kernel function. The popular choices for SVM kernels are a) Polynomial kernel, b) Sigmoid kernel and c) Gaussian kernel.

Polynomial Kernel is defined as

$$k(x, z) = (x^T z + r)^d \tag{2.66}$$

where d is the degree and r is the bias. Polynomial kernels are generally a good choice when we have some prior information about the data. It has two parameters d and r to be tuned and thus including the regularization parameter C, grid search would be a 3 dimensional grid search.

Sigmoid Kernel is defined as

$$k(x, z) = \tanh(px^T z + q) \tag{2.67}$$

Sigmoid Kernel has two parameters to tune and thus the grid search for SVM should be performed on a 3 dimensional grid. Also [72] has given a theoretical explanation that sigmoid kernel in general is not a better choice than the Gaussian Kernel.

Gaussian Kernel Gaussian Kernel is the most popular kernel for SVMs, specially in the cases when we don't have much prior information about the data. It is defined as

$$k(x, z) = \exp(-\gamma(||x - z||^2)) \tag{2.68}$$

The tunable parameter γ is the inverse width parameter. Gaussian kernel has just one parameter γ to tune and thus the grid search is just a 2D grid search. Due to the above mentioned reasons we decided to use the Gaussian kernel in our work.

The grid search for Gaussian kernel SVMs is performed for selecting the optimum combination of regularization parameter C and inverse kernel width parameter γ.

In the next two chapters we describe further, the usage of SVM in our work.

Chapter 3

Cascaded Classifier for Large-scale Data Applied to Automatic Segmentation of Articular Cartilage

.

Abstract

Many classification/segmentation tasks in medical imaging are particularly challenging for machine learning algorithms because of the huge amount of training data required to cover biological variability. Learning methods scaling badly in the number of training data points may not be applicable. This may exclude powerful classifiers with good generalization performance such as standard non-linear support vector machines (SVMs). Further, many medical imaging problems have highly imbalanced class populations, because the object to be segmented has only few pixels/voxels compared to the background. This article presents a two-stage classifier for large-scale medical imaging problems. In the first stage, a classifier that is easily trainable on large data sets is employed. The class imbalance is exploited and the classifier is adjusted to correctly detect background with a very high accuracy.

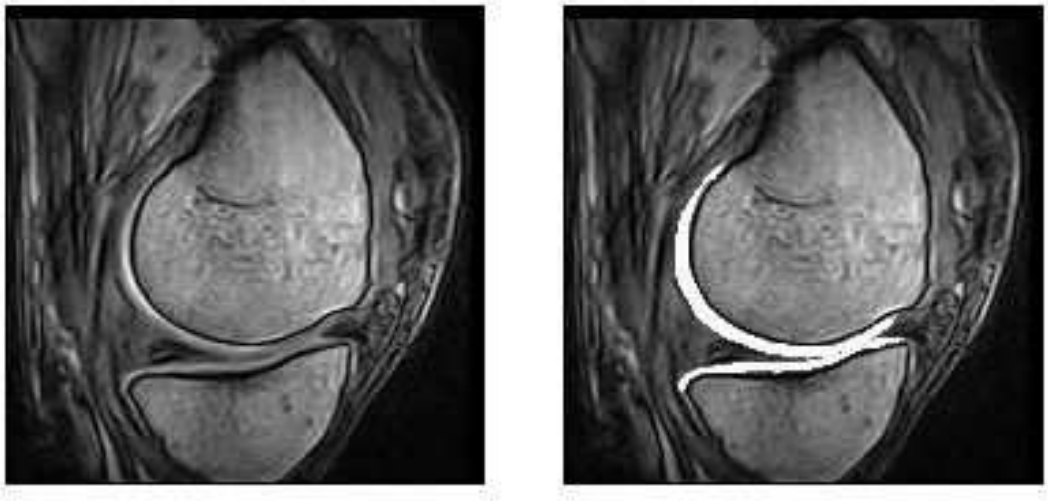
This chapter is based on "Cascaded Classifier for Large-scale Data Applied to Automatic Segmentation of Articular Cartilage", published in the proceedings of SPIE Medical Imaging 2012: Image Processing Conference [54]

Only the comparatively few data points not identified as background are passed to the second stage. Here a powerful classifier with high training time complexity can be employed for making the final decision whether a data point belongs to the object or not. We applied our method to the problem of automatically segmenting tibial articular cartilage from knee MRI scans. We show that by using nearest neighbour (kNN) in the first stage we can reduce the amount of data for training a non-linear SVM in the second stage. The cascaded system achieves better results than the state-of-the-art method relying on a single kNN classifier.
Keywords: large-scale data classification, image segmentation, cascaded classifier, two-stage classifier, support vector machines, osteoarthritis, magnetic resonance imaging (MRI), articular cartilage segmentation

3.1 Introduction

Segmentation of anatomical structures is one of the most important tasks in medical image analysis. It is often addressed by applying machine learning methods to pixel/voxel based classification. In general, the success of this approach depends on three ingredients. First, a good set of image features discriminating well between the anatomical structure and the background is required. Second, a sufficiently large amount of training data needs to be available to cover as much biological variability as possible. Third, the learning algorithm must be strong enough to give good segmentation results using the training data. This study focuses here on the latter two aspects. The huge amount of required training data limits the choice of the machine learning technique. Some powerful methods with good generalization performance and desired theoretical properties, such as standard non-linear support vector machines (SVMs[46, 63]), scale badly with the number of training data points and hence are not directly applicable. Therefore, we present a simple two-stage system that allows us to make use of such complex learning techniques for typical medical imaging tasks in order to achieve better classification and segmentation.

Osteoarthritis is one of the most common reasons for causing work disability,



(a) Slice of a knee MRI **(b)** Corresponding segmentation

Figure 3.1: Knee MRI slice and its segmentation into articular cartilage and background by a radiologist.

especially in the elderly population and has a large social and economic impact [73, 74]. MRI scans are most commonly used for noninvasive assessment of the articular cartilage [75]. The deterioration of the articular cartilage can be diagnosed using quantitative MRI analysis. We applied our approach to the segmentation of articular cartilage in low-field knee MRI scans (see Figure 3.1), which is required for the analysis of osteoarthritis. Our method is not limited to the tibial articular cartilage, and we will apply it for segmenting the femoral articular cartilage in future work.

3.1.1 Dataset

In this study we use low-field MRI scans. Low field MRI scanners have poorer image resolution and worse image quality than the high-field scanners. They are however much more cost effective than high-field MRI scanners, with lower installation and maintenance cost. The comfort level of patients is also higher with no claustrophobic feeling. Low-field MRI scans for quantitative analysis of articular

cartilage can be helpful for reducing the cost of clinical studies.

MRI was performed with an Esaote C-Span lowfield 0.18T scanner dedicated to imaging of extremities yielding a sagittal Turbo 3-D T1 sequence. Approximate acquisition time is 10 min with each scan having 104-116 slices of size 256×256. The spatial in-plane resolution of the scans are $0.70 \times 0.70\, mm^2$, with a distance between slices ranging between 0.70 - 0.94 mm, where the most common distance is 0.78 mm. 25 scans were used for training and a hold-out set of 114 scans were used for evaluation.

3.1.2 Related Work

Cartilage segmentation by radiologists is often done slice by slice (Figure 3.1), and this is a tedious and very time consuming task. Moreover these segmentations are found to have high inter- as well as intra-observer variability. Thus automated or semi-automated methods are desirable and computer-aided segmentation of the articular cartilage from the knee MRI scans is active research field. Methods can be divided into 2D and 3D approaches, depending on whether they rely on 2D methods to segment one slice at a time or directly use 3D segmentation. For instance, B-spline snakes were used by Stammberger et al. [76] to segment each slice. Lynch et al. [77] also developed a 2D approach using active contours. Active shape models for slice-by-slice cartilage segmentation, were used by Solloway et al. [78]. Assuming the bones are already segmented, Pakin et al. [79] used a region growing technique with clustering to segment the cartilage. Grau et al. [80] used watershed based approach but the method required 5-10 minutes of human intervention, to select the markers before segmenting a scan. User performed interactive registration of a knee template to a test scan has also been used [81, 82]. Folkesson et al. [71] used a very efficient, robust and fully automatic method to segment the cartilage in 3D and their method can be considered as the state-of-the-art method for fully automatic segmentation of articular cartilage. Bae et al. developed a semi-automated method based on a graph-cuts algorithm for segmentation and volumetric measurements of the cartilage from high-resolution knee magnetic resonance (MR) images from the Osteoarthritis Initiative (OAI) database and as-

sessed the intra and inter-observer reproducibility of measurements obtained via their method [83]. Chang et al proposed a semi-automatic segmentation method of knee cartilage based on radial transformation [84]. A method to simultaneously segmenting the bone and cartilage surfaces of a knee joint in 3D was presented by Yin et al [85]. Seim et al. presented a fully automatic method for segmenting bones and cartilages from MRI scans [86]. They used statistical shape model and graph-based optimization to firstly reconstruct femoral and tibial bone surfaces and then strating from bone surfaces they simultaneosly segment the cartilage using prior knowledge on the variation of cartilage thickness. Vincent et al. presented a fully automatic model based system for segmenting bone and cartilage in magnetic resonance (MR) images of the knee [87]. Their segmentation method is based on Active Appearance Models built from manually segmented examples from the Osteoarthritis Initiative database. Hinrichs et al. proposed a segmentation algorithm based on level sets for the 3D segmentation of knee articular cartilage [88]. The method incorporates non-linear diffusion for efficient image denoising. Fripp et al proposed a scheme consisting of three stages; automatic segmentation of the bones, extraction of the bone cartilage interfaces and segmentation of the cartilages [89]. Dodin et al. also developed an automatic segmentation algorithm for MRI scans obtained using 3T scanner and a knee coil. The imaging uses a double echo steady state (DESS) sequence, which contrasts cartilage and soft tissues including the synovial fluid [90]. Their algorithm was developed on 3-D images in which the bone cartilage interface for the femur and tibia was segmented by an independent segmentation process, giving a parametric surface of the interface.

In this thesis, we compare our methods with Folkesson et al. [71]. However, below, we discuss some other recent and state-of-the-art works on knee cartilage segmentation.

Lee et. al. [91] propose a fully automatic method in which they segment bone, bone-cartilage interface and cartilage, with the main contribution being cartilage segmentation. They first segment the bone using a modified version of branch and min-cut algorithm. After segmenting the bone they classify bone voxels to segment the bone cartilage interface (BCI) using two classifiers which are based on position

and local appearance. Finally, they use the BCI segmentation to designate the ROI for cartilage segmentation and segment the cartilage using localized Markov random fields. They evalute their method on 10 subjects.

Shan et. al. [92] propose multi-atlas approach to segment femoral and tibial cartilage from knee MRIs. They first use a simple model for estimating likelihood for femur and tibia bones. Then they apply multiatlas registration followed by label fusion to segment the bone. For cartilage segmentation, they incorporate shape prior based on multi-atlas approach and cartilage likelihood obtained from probablistic kNN. They validate their method on 18 subjects.

Wang et. al. [93] propose a method which requires pre-segmentation of bones. However they don't rely on explicit classification of BCI. Instead they compute distance features from each voxel to anatomical landmarks on the bone surface. Then they use iterative discriminative classifiers where probability maps generated by first pass are used to generate semantic contextual features for the second pass. They evaluate their method on 176 volumes of Osteoarthristis Initiative dataset (OAI).

Fripp et. al. [94] segment a hierarchical segmentation scheme where bones, which are easier to segment are segmented first segment bone using 3D active shape model initialized using affine atlas registration. Then they extract the BCI using prior knowledge about points belonging to the bone cartilage interface and image information. They further refine the bone segmentation, BCI extraction and thickness profile through their cartilage segmentation algorithm. They incorporate localized estimate of tissue properties and classification while the bone cartilage interface and thickness maps from the training data are utilized to generate the thickness-variation's principal component model for the points on BCI. Number of modes constrained is for capturing 90% of the training data. These are then incorporated in a 3D active surface model approach for segmenting the cartilage.

Yin et al. [95] propose a simultaneous segmentation method for multiple interacting surfaces of multiple interacting objects based on layered optimal graph segmentation. They call their method "LOGISMOS". They apply their method for the bone and cartilage segmentation in MRI scans of human knee.They train

their method on 9 scans and evaluated their method using leave one out test.

The main difference between our method and the methods mentioned above are that we segment the cartilage directly without using any prior segmentation of bones or extraction of BCI. Moreover, our approaches are purely a voxel classification approaches, whereas the above mentioned approaches are hybrid approaches which involve other techniques as well.

The rest of this chapter is organized as follows. In the next section we discuss shortly a general methodology for two stage classifier. Then in Section 3.3 we apply it to the automatic segmentation of tibial cartilage. The first stage is performed by a k-NN classifier while the second stage uses a SVM classifier. In particular, data as well as model selection are discussed. Then in Section 3.4 we present our results and evaluation of the proposed method. Finally we summarize and conclude in Section 3.5

3.2 Two Stage Classifier

In this section, we describe the general idea of our two-stage classifier. The approach is inspired by the cascaded object recognition architecture by Viola and Jones [96, 97].

Let us assume a binary classification problem. Further, we assume that the two classes have highly imbalanced populations with the background (negative class) data points outnumbering the class to be segmented (positive class) by at least an order of magnitude. Let X and $Y = \{-1, 1\}$ denote input and label space, respectively. First, we learn a hypothesis $h_1 : X \mapsto Y$ using all the ℓ samples in our training data set $D_{\text{train}_1} \subset (X \times Y)^\ell$. The learning algorithm at this stage is chosen to be able to handle a large ℓ and to produce a hypothesis with high sensitivity (recall rate). Ideally, it should identify all positive examples correctly while identifying as much members of the negative class as possible. This stage filters out points belonging to the background class in order to reduce the input for the subsequent classifier. All data points classified as non-background by h_1 are passed to the second stage. In this stage, we generate a hypothesis h_2 using the training data $D_{\text{train}_2} = \{(x, y) \,|\, (x, y) \in D_{\text{train}_1} \wedge h_1(x) = 1\}$. If $|D_{\text{train}_2}| \ll \ell$,

we then can afford to employ a classifier with less favorable scaling behavior with respect to the number of training data points than in the first stage and we are now less limited in the choice of the learning algorithm and pick the one we expect to result in maximum accuracy. The final two-stage classifier is given by

$$h(x) = \begin{cases} -1 & \text{if } h_1(x) = -1 \\ -1 & \text{if } h_1(x) = 1 \text{ and } h_2(x) = -1 \\ 1 & \text{otherwise} \end{cases} . \tag{3.1}$$

As we assume the fraction of data points belonging to the negative class to be much higher than the fraction of points belonging to the positive class, we expect that only a small fraction of data points must be handled by the second stage despite the high sensitivity of h_1. Therefore, the cascaded classifier inherits the applicability to large-scale data from the algorithm applied in the first stage and the high accuracy of the more complex classifier in the second stage.

3.3 Automatic Segmentation of Tibial Cartilage

We now describe how the cascaded strategy discussed above has been applied to tibial cartilage segmentation. A good cartilage segmentation system requires large-scale machine learning, not only because of the number of voxels in a single MRI scan but also because many of these scans are needed to cover sufficiently the biological variability. Our new approach allows to apply more complex classifiers and accordingly leads to a better segmentation. Our first stage uses a k-NN based rule, while the second stage uses a SVM rule. Two-stage architectures with nearest neighbor classification followed by an SVM have been proposed before (e.g., by Batra et al. [98]), but to our knowledge with different motivations and not in the domain of medical imaging.

The following subsections discuss all the important aspects of our two stage classifier.

3.3.1 Features

The classification is feature based, and we have used the same set of input features as Folkesson et al. did in [71]. We describe them now.

Two of the candidate features are position and intensity. Intensities are computed at different scale by convolution with Gaussians in a scale-space framework [99], three different scales are used to produce intensity based candidate features. Gaussian derivative features have also been used, they have been obtained by convolving the image with with derivatives of Gaussian, up to order three with respect to the spatial variables, so that intensity plus derivative features gives a complete description of the geometry up to order three, at three different scales. A next set of features are the eigen-values and the eigen-vectors of both the 3D Hessian matrix and and the structure tensor at three different scales. The features mentioned till now are the features that examine the first and second-order structures. Third-order tensors in the gradient direction on three scales are also included as candidate feature to examine the local third order structure.

Folkesson et al.'s candidate features [71] are thus: intensity, position, the three-jet, eigenvalues, and eigenvectors of both the Hessian and the structure tensor and the third-order tensor in the gradient direction. Every feature is calculated at three different scales (except position). Features (except intensity) were coupled 3 by 3 to allow them the same chance of getting picked during feature selection. Finally 36 features were selected using feature selection out of the above mentioned 178 features.

3.3.2 Training Data

We used the scans of 25 patients manually segmented by a radiologist. As mentioned earlier, each full scan consisted of approximately 104-116 slices, each slice being 256×256 in size. They are the same scans that were used to generate the training data for the state-of-the-art method of Folkesson et al. From each scan, a region of interest (ROI), covering about 30% of the volume of the scan was extracted. The ROI fully contain the cartilage volume. Same as [71], we include all the cartilage voxels in the training data-set, while the background voxels were

sampled very densely near the cartilage and rarely far from it. This sampling scheme is chosen because the voxels near the cartilage are much more difficult to classify when compared to the voxels far from the cartilage. Thus the idea was to increase their representation in the training data. Overall, we had approximately 2 million training voxels in the training data set.

3.3.3 Stage One

In our method, we used a classifier similar to the one by Folkesson et al. in the first stage but with a different purpose. The k-NN based classifier of Folkesson et al. produces for each voxel a posterior probability of belonging to a class as the fraction of k-nearest neighbors that belong to that class. The training of the classifier consisted in reducing the number of features (to 36) and posterior threshold θ that optimizes the segmentation result. The value of k used by Folkesson et al was k=100. We used the same value of k in the our first stage. As our first stage differs in its objective, which is to reach 100% sensitivity (i.e. zero false negative) and as much as possible true positives, it leads to a different threshold.

3.3.4 Stage Two

In the second stage, we employed a non-linear soft-margin Support Vector Machine [46]. We provide a brief description of SVM and how to select the model parameters in our problem.

3.3.5 Support Vector Machines

Support vector machines are state-of-the-art in binary classification. Non-linear SVMs transfer the input data into a large-dimensional feature space and perform linear classification in that space. Given a positive semi-definite kernel function $k : X \times X \to \mathbb{R}$, one considers the feature space generated by this kernel, $\mathcal{H}_k = \text{span}\{k(x, \cdot) \,|\, x \in X\}$ and the class of affine functions on it, $\mathcal{H}_k^b = \{f = g + b \,|\, g \in \mathcal{H}_k,\ b \in \mathbb{R}\}$. The decision boundary induced by the sign of a function $f \in \mathcal{H}_k^b$ is a hyperplane in $\mathcal{H}_k$. Given training data $(x_i, y_i) \in X \times \{-1, 1\}$, $i = 1, \ldots, \ell$, a

1-norm soft margin SVM[46] finds a solution to the *regularized risk minimization problem*

$$\underset{f\in\mathcal{H}_k^b}{\text{minimize}} \quad C\sum_{i=1}^{\ell} L_{\text{hinge}}(y_i, f(x_i)) + \frac{1}{2}\|f\|^2$$

with loss function $L_{\text{hinge}}(y, f(x)) = \max\{0, 1 - yf(x)\}$. The parameter $C > 0$ controls the trade-off between reducing the empirical loss L_{hinge} and the complexity of the hypothesis $\|.\|$ measure by the norm of its $\mathcal{H}_k$ component.

The SVMs in this study use standard Gaussian kernels on the full 178 dimensional (without feature selection) image-feature.

$$k_\gamma(\mathbf{x}, \mathbf{z}) = e^{-\gamma(\|\mathbf{x}-\mathbf{z}\|)^2}, \quad \mathbf{x}, \mathbf{z} \in X = \mathbb{R}^{178}.$$

Model Selection To adjust the hyper-parameters of the SVM classifier, the kernel bandwidth parameter γ and the regularization parameter C, we performed grid searches. We determined an initial estimate γ_{J} for the bandwidth parameter using the heuristic proposed by Jaakkola et al. [100], which provides a reasonable initial guess for the bandwidth parameter of a radial Gaussian kernel. The heuristic considers all pairs consisting of an training input vector from the positive class and a training input vector from the negative class; computes the difference in input space between all pairs; and assumes that the median of these distances can be used as a measure of scale. More formally, we compute

$$\gamma_J = \frac{1}{2\,\text{median}(\{\|\mathbf{x}_i - \mathbf{x}_j\| \mid (\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j) \in D_{\text{train}} \wedge y_i \neq y_j\})^2} \ . \tag{3.2}$$

We employed the LIBSVM software by Chang and Lin for non-linear SVM training [101].

For the cross validation we divided the 25 scans set into 5 subsets, each having 5 scans. We perform 5 iterations for cross validation with each time a different subset chosen as validation set and the rest four are jointly the training set. The above approach of cross-validation was employed to reduce the number of hyper-parameter combinations and thus making the cross-validation faster. Then we picked the hyper-parameter pair (C', γ') minimizing the cross-validation error with

the input hyper-parameters to LIBSVM being

$$C' \in \{2^{-11}, 2^{-10}, \ldots, 2^{17}\} \text{ and } \gamma' \in \{\gamma_{\mathrm{J}} \cdot 2^{-7}, \gamma_{\mathrm{J}} \cdot 2^{-2}, \ldots, \gamma_{\mathrm{J}} \cdot 2^{5}\} \ .$$

After selecting the best hyper-parameter pair once, we placed a narrow-grid (3-by-3) centered at (C', γ'). At this stage we introduced weight ratio parameter $W \in \{1.0, 1.1, 1.2, \ldots, 1.8\}$, which makes it possible to select different regularization parameter for the two classes. So, the regularization parameter for the cartilage and the background class eventually become $C \cdot W$ and C respectively. Then we performed cross-validation for each combination of C, γ and W to get the final parameters. The final training time was approximately 6 days on 2.93 GHz processor and 4 GB RAM system.

3.4 Evaluation and Results

We tested our method on a set of 114 unseen scans. For fast evaluation, for each scan we chose the background test points by randomly sampling very densly near the cartilage and rarely far for from it, with the sampling probabilty varying linearly. However, for each scan, all the cartilage points were taken as foreground test points. We also evaluated the state-of-the-art [71] method on the same test scans to compare the two methods. Sampling on the background of each test scan was performed just once and then the same sampled points were used for comparison.

In our experiments the first stage filtered out more than 86 % of the training and test data. We evaluated the segmentation based on the dice similarity coefficient (DSC) given by

$$\mathrm{DSC}(A, B) = \frac{2(|A \cap B|)}{|A| + |B|}$$

where A and B are manual and automatic segmentations, respectively [102]. Table 3.1 shows the mean and standard deviation of the dice similarity coefficient, accuracy, sensitivity and specificity obtained over testing 114 scans using the proposed method and the state-of-the-art method. As it can be seen our method

Table 3.1: Comparison of classifiers applied for cartilage segmentation in MRI. The abbreviation DSC. stands for the dice similarity coefficient, Acc. for accuracy, Sens. for sensitivity, Spec. for specificity. The proposed cascaded kNN+SVM classifier is referred as Two-stage.

Classifier	Over 114 Scans	DSC	Acc.	Sens.	Spec.
Two-stage	Mean	0.8432	98.3074	85.2480	99.0741
	Std. Dev.	0.0428	0.4153	8.6414	0.4180
Folkesson et. al. [71]	Mean	0.8329	98.2000	83.6139	99.0594
	Std. Dev.	0.0425	0.4614	8.3703	0.4692

(a) Tibial Cartilage segmented from a slice by the Radiologist

(b) Segmentation obtained by two stage kNN+SVM

(c) Segmentation obtained by one-stage kNN

Figure 3.2: An example knee MRI slice where two-stage kNN+SVM performs clearly better than the one-stage kNN method

gets higher mean for DSC, accuracy, sensitivity and specificity. The differences in accuracy and DSC between the one-stage and the two-stage approach are both statistically significant (Wilcoxon rank sum test, $p < 0.05$). For our test data set of background sampled scans (having average of 80000 voxels per scan) this comes at the reasonable cost of a 10%-13% increase in time for classification over 1-stage of kNN, because around 14 % of the data have to be processed by both classification stages. The total time taken on 2.8 GHz processor and 4GB RAM was 42 minutes. However, for a full scan (no background sampling, 2 million voxels in the ROI),

as the classification time was too long for stage one kNN screeening, as done by Folkesson et al., [71], approximate nearest neighbor framework developed by [103] with $\epsilon = 2$ was used. ϵ is a parameter such that the ratio of distance between a nth ($n \leq k$) reported neighbor and the true nth neighbor can't be more than $1+\epsilon$. The time taken by stage-one was 85 minutes and the total time taken for full scan by 2-stage classifier was 195 minutes.

We compared to with other classification rules, Linear SVM, logistic regression and another kNN at second stage (two-stages kNN). Overall, while the proposed method (two-stage kNN + SVM) significantly outperformed the state-of-the-art one-stage kNN [71], both methods performed clearly better than the linear SVM and logistic regression.

In that respect the large-scale machine learning software LIBLINEAR [104] was used for the linear SVM and logistic regression. The regularization parameters of the linear SVM and logistic regression were adjusted using grid-search.

When using the two-stages classifier with kNN in the second stage, the results become worse than the original method. However it gave better results than the linear classifiers (results not shown). Figure 3.2 shows a slice segmented by the radiologist, our method and the one-stage kNN. Although, the average DSC obtained for Folkesson et al. [71] was 83.29%, the slice shown in the figure is taken from a scan for which our method clearly outperforms one-stage kNN. Figure 3.3 shows one of the very few cases where one-stage kNN performed slightly better than our method. The segmentation suggests that the use of shape modeling as a post processing step can increase the overall segmentation performance. For better illustration, we have shown a slice taken from a 3D segmentations and its worth mentioning again that none of the two methods are 2D segmentation method. However, manual segmentations are performed in a 2D way(slice by slice).

We also evaluated interscan segmentation reproducibility on 31 pairs of scans, each pair obtained within a week. There was no registration performed between the pairs of scans. The SVM model obtained from model selection for segmentation task was used to evaluate the inter-scan reproducibility. There was no overlap between the training and the test patients. The same radiologist segmented both

(a) Tibial Cartilage segmented from a slice by the Radiologist

(b) Segmentation obtained by two stage kNN+SVM

(c) Segmentation obtained by one-stage kNN

Figure 3.3: One of the few examples of knee MRI slice where one-stage kNN performs slightly better than the two-stage kNN+SVM method

Table 3.2: Comparing interscan tibial cartilage segmentation reproducibility on 31 pairs of scans

Method	**RMS-CV**
Two-stage	0.0597
Folkesson et. al. [71]	0.0700
Manual	0.1030

the scans of each pair. The evaluation was based on the *RMS-CV* score calculated as follows. Let V_s^i and V_r^i be cartilage volumes obtained from scan and re-scan of the ith pair. We calculate the coefficient of variation for ith pair

$$C_v^i = \frac{\sqrt{2}(|V_s^i - V_r^i|)}{V_s^i + V_r^i}$$

which is the same as the ratio of the standard deviation to the mean of two volumes. The *RMS-CV* score is given as $\sqrt{\sum_{i=1}^{31} {C_v^i}^2/31}$. The lower the *RMS-CV* score, the better the reproducibility. Table 3.2 summarizes the results obtained, showing that our method performed better than the radiologist as well as the state-of-the-art method.

3.5 Conclusion

We present a general approach that allows to use complex non-linear classifiers with good generalization ability for large-scale medical imaging problems under the assumption of highly unbalanced class frequencies. We show that by applying our method to MRI scans of the knee, we can improve on the current state-of-the-art in articular cartilage segmentation. This is a relevant medical imaging task supporting the analysis of osteoarthritis.

The two-stage cascaded system is a simple and flexible approach to improve the performance of classifiers for large-scale problems. It is inspired by the object recognition architecture by Viola and Jones [96, 97]. The basic assumption of highly unbalanced class frequency is often met in medical applications, especially in image segmentation with large background parts and in general when trying to identify a few diseased persons in a large population. Thus, the approach is particularly well-suited for medical imaging problems. The architecture is very flexible, the type of classifiers and even the number of stages can be adapted to the problem at hand.

We were able to improve the segmentation of knee cartilage from in MRI scans. This required processing large amounts of data and our experiments showed that linear classifiers, typically employed for large-scale machine learning, did not provide sufficiently good segmentation accuracy. Our two-stage procedure significantly reduced the number of data points passed to the second classifier. Therefore we could employ a non-linear SVM at the final stage. The reduced training set size did not only allow for proper training and model selection of the SVM in reasonable time, it also led to an SVM classifier with faster execution time, because the number of support vectors and therefore the evaluation time of h_2 scales linearly in the number of training pattern in our scenario [59].

Of course, in a two-stage process some of the inputs have to be classified twice. Further, both the storage requirements as well as the execution time of the nearest neighbor classifier scales linearly with the number of training data points (if no approximation scheme [103] is used). In the current system, the training of the non-linear SVMs in the model selection procedure was the most time consuming

part. The scaling of standard SVM training the main motivation for the two-step architecture. However, computing a close to optimal solution to the SVM optimization problem (see Section 2) may not be necessary for good classification results. Instead, we could try to find a good approximation to the SVM solution quickly. This can be achieved by *online* SVMs.[105, 106] In future work, we will evaluate online SVMs for the articular cartilage segmentation problem. Fast approximate training may render the first nearest neighbor classification stage unnecessary.

Chapter 4

Femoral Cartilage Segmentation in Knee MRI Scans Using Two Stage Voxel Classification

Abstract

Using more than one classification stage and exploiting class population imbalance allows for incorporating powerful classifiers in tasks requiring large scale training data, even if these classifiers scale badly with the number of training samples. This led us to propose a two-stage classifier for segmenting tibial cartilage in knee MRI scans combining nearest neighbor classification and support vector machines (SVMs). Here we apply it to femoral cartilage segmentation. We describe the similarities and differences between segmenting these two knee cartilages. For further speeding up batch SVM training, we propose loosening the stopping condition in the quadratic program solver before considering moving on to other approximation techniques such as online SVMs. The two-stage approach reached a higher accuracy in comparison to the one-stage state-of-the-art method. It also achieved better inter-scan

This chapter is based on "Femoral Cartilage Segmentation in Knee MRI Scans Using Two Stage Voxel Classification" by Adhish Prasoon, Christian Igel, Marco Loog, Francois Lauze, Erik Dam, and Mads Nielsen, published in the proceedings of 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC 2013) [55]

segmentation reproducibility when compared to a radiologist as well as the current state-of-the-art method.

Keywords: femoral cartilage, support vector machine, nearest neighbor classifier, online support vector machine, osteoarthritis, magnetic resonance imaging

4.1 Introduction

The need to cope with large scale data in medical imaging often limits the use of complex classifiers having excellent generalization ability. An example of such a classifier is a non-linear support vector machine (SVM, [46]), where the training time scales worse than quadratically with the number of training data points. In our previous work [54], we presented a two-stage cascaded classifier approach to overcome this restriction. The proposed classifier was applied to segment tibial cartilage in low-field knee MRI scans and outperformed the state-of-the-art method. As a step towards completing the study we apply the similar approach for segmenting femoral cartilage, also from low-field knee MRI scans. The segmentation of articular cartilage is useful for the quantitative analysis of the deterioration of articular cartilage, which causes osteoarthritis. Osteoarthritis is one of main causes of work disability through out the world specially for the elderly population. Non-invasive assessment of articular cartilage are most commonly done using MRI scans [75].

In a general two-stage classifier for segmentation, we have a classifier trainable on huge data-sets in the first stage. However, the goal of this first stage is not necessarily to achieve best segmentation results, but to maximize sensitivity, that is, to minimize false negatives. The points classified as background by the first stage are labeled accordingly, while all the points classified as foreground go through a second stage of classification. The classifier used at this stage can be a more powerful classifier, which may scale badly with number of training data points. However, if the background population is large compared to the foreground population and a large portion of background population is screened in the first stage, a significantly smaller portion of data points is fed into the second-stage classifier.

This makes it possible to use classifiers scaling badly with number of training data points.

In this study, we extend our earlier work and apply the two-stage approach to segmenting femoral cartilage. The approach is compared to the state-of-the-art method that is based on one stage of nearest neighbor classification. We discuss the similarities and differences in segmenting femoral and tibial cartilages as well as the challenges faced due to the even higher amount of training data compared to [54]. Furthermore, we consider images of subjects scanned twice within one week and investigate the inter-scan reproducibility of the proposed classifier in comparison to a radiologist and the current state-of-the-art method.

4.2 Related Work

Computer-aided segmentation of the articular cartilage from the knee MRI scans is an active research field. Methods either rely on 2D approaches to segment slice by slice or directly use 3D segmentation. Stammberger et. al. [76] used b-splines to segment each slice of MRI scan. Another slice-by-slice cartilage segmentation method based on active shape models was proposed by Solloway et. al. [78]. Folkesson et. al. [71] developed a 3D voxel classification approach which can be considered as state-of-the-art method for fully automatic segmentation. A semi-automated method was developed by Bae et. al. [83]. Their segmentation method is based on graph-cut algorithm. They performed volumetric measurements of the cartilage from high-resolution knee magnetic resonance (MR) images from the Osteoarthritis Initiative (OAI) database and assessed the intra and inter-observer reproducibility of measurements obtained via their method. A semi-automatic method based on radial transformation was proposed by Chang et. al. [84]. Yin et. al. proposed a method for simultaneous segmentation of bone and cartilage surfaces [85]. A fully automatic method was proposed by Seim et. al. who segmented bones and cartilage from MRI scans using statistical shape model and graph based optimization [86]. Vincent et. al. presented a fully automatic system [87] based on active appearance models. A three stage scheme was proposed by Fripp et. al. [89]. In the first stage automatic segmentation of the bones is

performed. In the second stage, the bone cartilage interfaces is extracted. In the final stage segmentation of the cartilages is performed. Dodin et. al. proposed automatic segmentation method for knee MRI scans acquired using 3T scanner and a knee coil [90]. They segmented bone cartilage interface for tibia and femur independently. A level sets based algorithm was proposed in [88] for 3D segmentation, and [95] incorporated multiple spatial inter-relationship on n-dimensional graphs followed by graph optimization that yields a globally optimal solution to segment cartilage.

In this thesis, we compare our methods with Folkesson et al. [71]. However, we have also discussed some other recent and state-of-the-art [91, 92, 93, 94, 95] works on knee cartilage segmentation (please see section 3.1.2).

4.3 Approach

This section presents the two-stage classifier and describes its application to the segmentation of femoral cartilage. We also consider the challenges associated with extending the study from tibial cartilage segmentation to femoral cartilage segmentation and comment on speeding up SVM training by approximating the SVM solution.

4.3.1 Two-stage Classifier

Let us assume w.l.o.g. a binary segmentation problem where the population of the positive class is less than the negative class population by at least an order of magnitude. Let X be the input space and $Y = \{-1, 1\}$ the output. A hypothesis $h_1 : X \mapsto Y$ is learned in the first stage using all the training data. Let ℓ be the number of samples and $D_{\text{train}_1} \subset (X \times Y)^\ell$ be the training data. The hypothesis h_1 is tuned to achieve maximum sensitivity and, thus, having minimum false-negatives. This stage should use a learning algorithm which can handle a very large number of training data points. The data points classified as background by first stage classifier are labeled as background and rest of the points $D_{\text{train}_2} = \{(x, y) \,|\, (x, y) \in D_{\text{train}_1} \wedge h_1(x) = 1\}$ are used to train our second stage hypothesis h_2 . The aim of the learning algorithm at this stage is to achieve good segmentation

performance. As the number of data points at this stage is just a small fraction of ℓ, we can employ a powerful classifier at this stage, even if it scales badly with training data population. This way, the final two-stage classifier has good generalization ability and can also handle huge training data sets. The two-stage classifier can be summarized as

$$h(x) = \begin{cases} -1 & \text{if } h_1(x) = -1 \\ -1 & \text{if } h_1(x) = 1 \text{ and } h_2(x) = -1 \\ 1 & \text{otherwise} \end{cases} .$$

Figure 4.1 depicts the general two stage classifier.

4.3.2 Automatic Segmentation of Femoral Cartilage

This section presents the application of a two-stage classifier to segmenting femoral cartilage. Whenever needed, we will also refer to tibial cartilage segmentation for comparison.

Training Data and Features Training data was extracted using 25 scans, which are exactly the same scans as used by state-of-the-art method [71]. Firstly, a region of interest from each MRI scan was extracted. The volume of region of interest is 30% of the volume of the MRI scan. Each MRI scan has around 6.85 million voxels with a region of interest of approximately 2 million voxels. As the background points are too high in number, we sample the background from the ROI and take all the cartilage voxels in our training data. The sampling is performed very densely in the region close to the cartilage, rarely in the region far from the cartilage and the sampling probability varies linearly between two values. For femoral cartilage, the number of all the cartilage points (from 25 scans) is 295,403 while the number of background points is 2,408,864. In our earlier study on tibial cartilage, the number of tibial cartilage and background points were 119,684 and 1,892,696 respectively. As we can see, femoral cartilage is considerably bigger than the tibial, resulting in higher number of cartilage as well as background data points. We use the same set of 178 features which were used

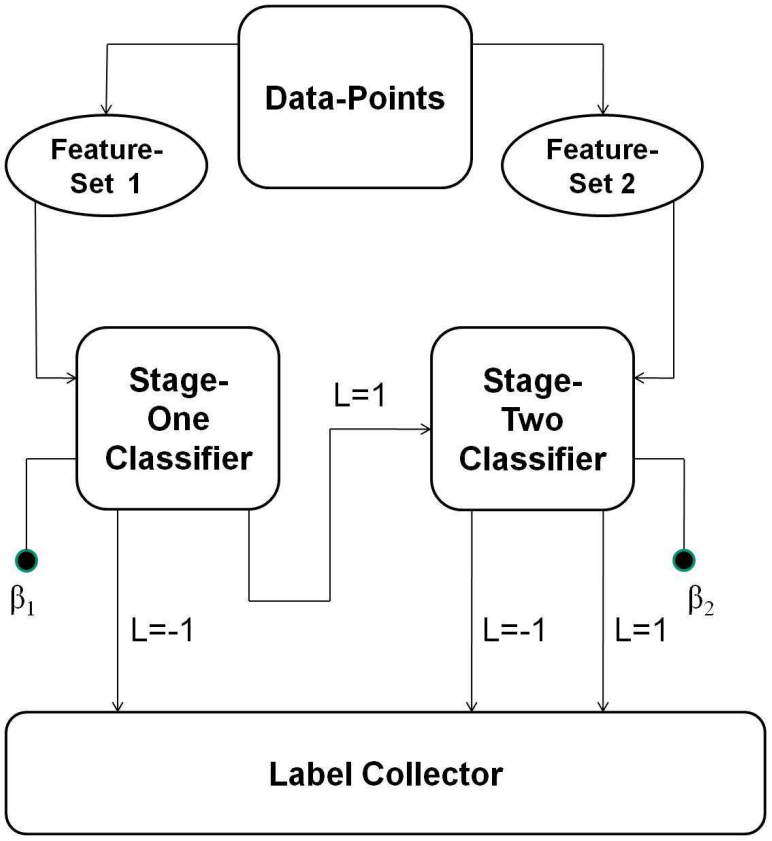


Figure 4.1: General concept of our two-stage classifier, where β_1 denotes parameters used to tune the first stage classifier for maximum sensitivity, while the parameters β_2 are used to tune the second stage for best segmentation performance. The labels $L = 1$ and $L = -1$ refer to cartilage and background voxels, respectively.

by Folkesson et. al. [71] as candidate features. Folkesson et. al. used features selection to find a smaller set of features in order to improve the performance.

The Two Stages for Femoral Cartilage Segmentation Stage one of our classifier is similar to the state-of-the-art one-stage kNN of Folkesson et. al. [71]. However, they have different aims. The one-stage kNN is trained to select the value of k, a smaller set of features using a feature selection method, and a posterior threshold t that deals with the large class imbalance to achieve best possible segmentation results. Let p_b be the posterior probability of a voxel being in back-

ground class, then the voxel is classified as background if $p_b > t$ and as cartilage otherwise. The features selected in case of tibial and femoral cartilage are slightly different. The number of selected features for tibial cartilage was 36 while that for femoral cartilage it was 42. The main difference between kNN used by [71] and our stage-one is the purpose of the classifier. We adjust t in order to achieve maximum sensitivity or minimum false positives. The value of k used by Folkesson et. al. was 100 for both the cartilage and we also use the same value of k.

Stage two of our classifier is an SVM with Gaussian kernel. We employed LIBSVM [101] for training the SVM. The training data comprised the points labeled as cartilage in stage-one. Although stage one used only a selected subset of features, in the second stage the SVM used all the 178 features. In case of the tibial cartilage, we performed nested grid search using cross-validation (splitting the available training data) as performance criterion. We searched for a good combination of kernel width parameter and regularization (commonly denoted by γ and C) on a 13×29 grid. After finding the best of these 377 combinations, we placed a second narrow 3×3 grid around the optimum value of C and γ. At this point we introduced a weight ratio parameter $W \in \{1.0, 1.1, 1.2, \ldots, 1.8\}$, which made it possible to select different regularization parameters. The final regularization parameters of cartilage and background class were $C \cdot W$ and C respectively.

Performing grid search in a big space of 377 combinations was time consuming even with a lot of computing resource. However, when performing the grid search for the second-stage of femoral cartilage segmentation, we placed just a 3×3 grid around the same pair (C, γ) which was found to be optimum during grid-search for the second-stage of tibial cartilage segmentation. The training dataset of same 25 patients was used as used in the case of tibial cartilage in the previous chapter. Also, the cross-validation was performed using same 5 folds as in the case of tibial cartilage. The good results that we achieved in case of femoral cartilage segmentation, using the hyper-parameters similar to what we learnt for tibial cartilage segmentation, show the robustness of our two-stage classifier.

4.3.3 Speeding-up SVM Training: Online Learning vs. Batch Learning with Low Accuracy

There were more than 700,000 training data points more in case of femoral cartilage than in case of tibial. On top of that, the first-stage of kNN performed slightly worse in the case of femoral cartilage segmentation when compared to tibial cartilage segmentation, thus the specificity achieved for maximum sensitivity was lower. Thus, the percentage of points screened in the first stage of femoral segmentation was lower than in the tibial case. In fact, the number of training data points for second stage SVM in case of femoral was 688,128, while in case of tibial the second stage SVM had to handle only 262,142 data points. Thus, model selection and final training gets very time consuming.

We consider non-linear SVMs. Training the machines amounts to solving a quadratic program (QP) having time complexity $\Omega(\ell^2)$ [60]. We use iterative sequential minimal optimization, and to speed up SVM training for the femoral cartilage classification, we loosened the stopping criterion from $\varepsilon = 0.001$ to $\varepsilon = 0.5$. We use the common stopping criterion as discussed, e.g., in [107]. Let $\alpha_1, \ldots, \alpha_\ell$ denote the coefficients of the SVM dual objective function f and let g_i denote the partial derivative of f with respect to α_i. Then we stop when

$$\max\left(\max_{\alpha_i<C, y_i=1} g_i, \max_{\alpha_i>0, y_i=-1} -g_i\right) - \min\left(\min_{\alpha_i<C, y_i=-1} -g_i, \min_{\alpha_i>0, y_i=1} g_i\right) \qquad (4.1)$$

falls below the threshold ε. After loosening the stopping criterion the training time on 2.93 GHz processor and 12GB RAM system was approximately 25 days.

An alternative to this approach is using an online SVM such as LASVM [108]. However, tuning ε is simpler, and we found it to produce more accurate solutions than LASVM within the same time budget in our application.

We also conducted experiments with LASVM to solve the problem in just one stage using all the 178 features and all the training data points. However, we observed no improvement in performance and too long training times. Indeed, the training did not complete even within a month.

Table 4.1: Comparison of classifiers applied for femoral cartilage segmentation. DSC stands for the dice similarity coefficient. The proposed cascaded classifier is referred to as two-stage 2-stage. All values are mean over 114 scans.

Classifier	DSC	Accuracy	Sensitivity	Specificity
2-stage	0.8115	96.3234%	80.8236%	98.0760%
Folkesson et. al. [71]	0.7984	96.0821%	79.7736%	97.8938%

4.4 Evaluation and Results

We tested our method on a set of 114 unseen scans. For fast evaluation, for each scan we chose the background test points by randomly sampling very densly near the cartilage and rarely far for from it, with the sampling probability varying linearly. However, for each scan, all the cartilage points were taken as foreground test points. We also evaluated the state-of-the-art [71] method on the same test scans to compare the two methods. Sampling on the background of each test scan was performed just once and then the same sampled points were used for comparison. We used Dice Similarity Coefficient to evaluate the segmentation performance,

$$\mathrm{DSC}(A, B) = \frac{2(|A \cap B|)}{|A| + |B|}$$

where A and B are manual and automatic segmentations. In table 4.1 we compare results obtained by our two-stage method with the state-of-the-art one stage kNN. Our method performed statistically significantly better than the one-stage kNN in terms of DSC and accuracy (Wilcoxon rank-sum test, $p < 0.05$), with both better sensitivity and specificity. For our test data set of background sampled scans (having average of 108000 voxels per scan), the testing time of our method was 30-35% more compared to the one-stage kNN for femoral cartilage. The average time taken by a 2 stage classifier was on 2.8 GHz processor and 4GB RAM was 78 mins. However, for a full scan (no background sampling, 2 million voxels in the ROI), as the classification time was too long for stage one kNN screening, as done by [71], approximate nearest neighbor framework developed by [103] with $\epsilon = 2$ was used. The average time taken by stage-one was 4 hours and the average time

taken for full scan by 2-stage classifier was 14 hours.

We also evaluated interscan segmentation reproducibility on 31 pairs of scans, each pair obtained within a week. There was no registration performed between the pairs of scans. The SVM model obtained from model selection for segmentation task was used to evaluate the inter-scan reproducibility. There was no overlap between the training and the test patients. The same radiologist segmented both the scans of each pair. The evaluation was based on the *RMS-CV* score calculated as follows. Let V_s^i and V_r^i be cartilage volumes obtained from scan and re-scan of the ith pair. We calculate the coefficient of variation for ith pair

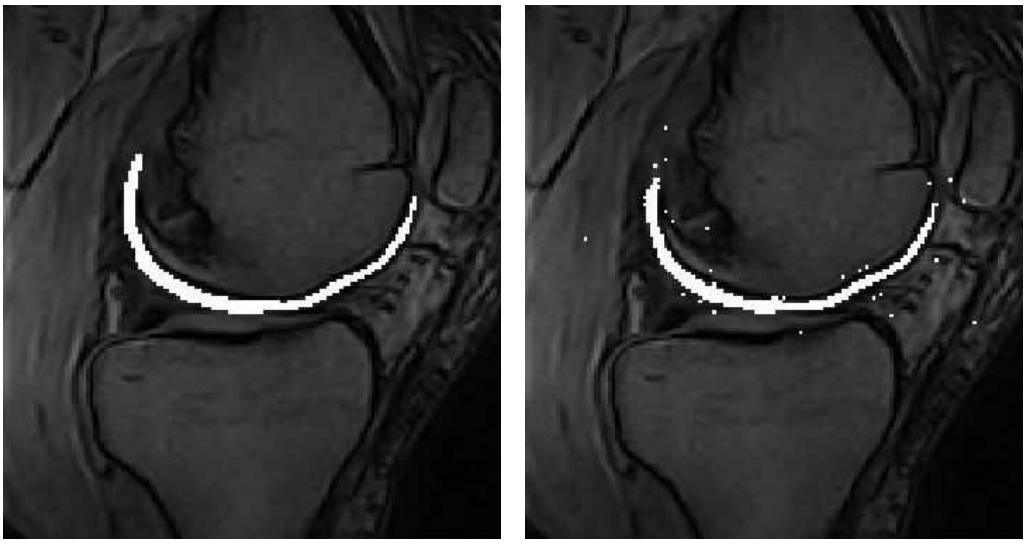
$$C_v^i = \frac{\sqrt{2}(|V_s^i - V_r^i|)}{V_s^i + V_r^i}$$

which is the same as the ratio of the standard deviation to the mean of two volumes. The *RMS-CV* score is given as $\sqrt{\sum_{i=1}^{31} {C_v^i}^2/31}$. The lower the *RMS-CV* score, the better the reproducibility. Table 4.2 summarizes the results obtained, showing that our method performed better than the radiologist as well as the state-of-the-art method.

Table 4.2: Comparing interscan femoral cartilage segmentation reproducibility on 31 pairs of scans

Method	**RMS-CV**
2-stage	0.0785
Folkesson et. al. [71]	0.0810
Manual	0.1140

Figure 4.2 shows a slice segmented by the radiologist and our method. A slice is taken from the 3D segmentation for visualization purpose (actually, radiologists segment the scans in a slice by slice manner). The resulting segmentations suggest that some post-processing can further increase the segmentation results.



(a) manual **(b)** two-stage

Figure 4.2: Slice taken from a 3D MRI scan segmented by (a) a radiologist and (b) our two-stage method. The slice was chosen to demonstrate that the segmentation can still be improved by (simple) post-processing.

4.5 Discussion

The proposed two-stage classification method is a general tool for scenarios in which the number of training data points is huge and the classes are unbalanced. These scenarios are often found in medical imaging applications and thus such a classifier is particular useful in this field. Its application for segmenting articular cartilage from low field knee MRI scans was very successful. The two-stage method outperformed the state-of-the-art one-stage kNN and also achieved better interscan segmentation reproducibility when compared to one-stage kNN and the manual segmentations done by a radiologist. For increasing the speed of the SVM training, we found no advantage in using online SVMs over simply reducing the accuracy of batch SVM training (by loosening the stopping condition in the quadratic program solver). Replacing the two-stage classifier by a single online SVM did not lead to better performance given our time budget.

Chapter 5

Convolutional Neural Network

Having a discriminative and powerful set of features is vital for any classification task. More commonly, we pre-define a set of features which seem to be appropriate for problem at hand. Also, very often, we perform feature-selection to find a smaller and optimum subset of features from a pre-defined larger set of features. However, instead of pre-defining a features set, the features can also be learnt autonomously in a data driven manner.

Before moving on to discuss convolutional neural networks, we would like to briefly discuss feedforward neural network and the problems associated with them.

5.1 Feed Forward Neural Networks

Before starting discussing feed forward neural network, we would like to describe the term "neuron" which is the basic entity of any neural network. A neuron as shown in figure takes several inputs and produces an output/activation. The weighted sum of these inputs are sent through an activation function to find the output of the neuron. Figure 5.1 depicts the process.

Part of this chapter is based on Convolutional Neural Network's introduction from "Deep Feature Learning for Knee Cartilage Segmentation Using a Triplanar Convolutional Neural Network" by Adhish Prasoon, Kersten Petersen, Christian Igel, Francois Lauze, Erik Dam, and Mads Nielsen, published in the proceedings of 16th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 2013). [56]. The part 5.5 on backpropagation is added to form this chapter.

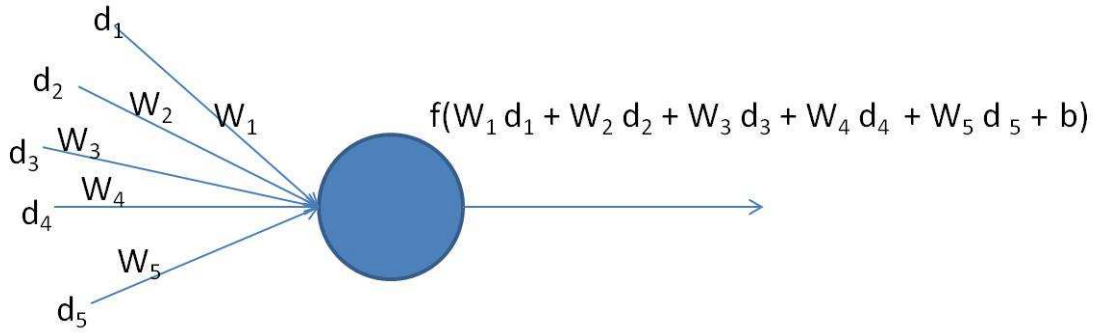


Figure 5.1: A neuron. Each input is multiplied by a weight and then summed up to be passed through an activation function

Research work in feedforward networks was started by Rosenblatt in 1962 [109]. A feed forward network has several neurons arranged in a layered structure (see figure 5.2). The first layer is called the input layer while the last is the output layer. The layer in between is called hidden layer. Although the example in figure (see figure 5.2) has single hidden layer, there can be multiple hidden layer too in a feed forward network. For calculating the output of the hidden layer neuron, the weighted sum of activation of all the input neurons is sent through a function, which is often a non-linear activation function such as sigmoid. Their is a unique weight for each pair of input layer's neuron and the hidden layer's neuron. Thus each pair of hidden layer neuron and input layer neuron are connected to each other. Now a weighted sum of each neuron of the hidden layer is given as input to the output neuron. The sum is then passed through activation function to get the final output. Although the figure shows single neuron in the output layer, there can be multiple neurons in the output layer as well.

In the feed forward neural network each neuron in a layer is connected to each neuron in the next layer, thus, as the number of hidden layers grow, the number of weights eventually become too large to handle. The other main disadvantage, specifically related to the field of computer vision is due the fact that each neuron pair of two subsequent layers has different weight. Thus, the weights learnt to

recognize an object in one part of the image can't be used to recognize the same object at a different location in the image.

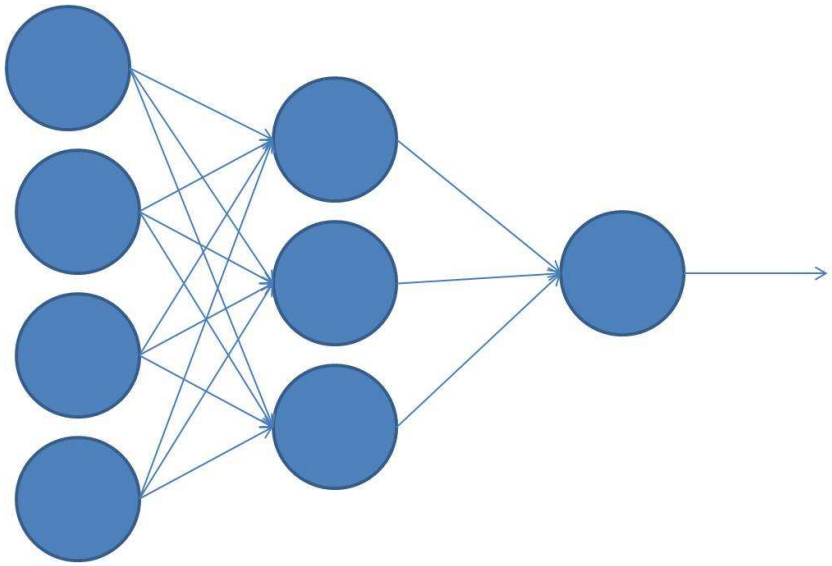


Figure 5.2: A feedforward neural network with one hidden layer and one output neuron. Each pair of two subsequent layer's neurons are connected through a weight

5.2 Convolutional Neural Network

Visual cortex which is responsible for processing visual information in the brain, has sequence of areas which process the information in a low to high abstraction level. The study of visual cortex shows that the neurons present in it get activated by stimuli generated by localized fields. Linear filtering in image processing is performed through convolution in spatial domain (or element-wise multiplication in frequency domain). However the idea behind CNN is to learn these filters in a data-driven manner. In machine learning, deep learning networks have multiple non-linear hidden layers and can represent the data in hierarchical way with lower to higher abstraction. Convolutional neural networks (CNNs) are variants of multilayer perceptron, which are inspired by visual cortex and have a deep learning architecture. The two main layers of CNN are convolutional layer and subsampling layer. Images which can be considered to be stationary have almost the

same statistics in the different smaller parts of the image. It also means that if we learn feature extractors for one part of the image, the same feature extractors can be used to extract features from other parts of the image. This idea is the main motivation behind the convolutional layer. However to reduce the computational complexity and to avoid over-fitting we need to summarize the features, which leads to the idea of having sub-sampling layer.

[47] applied CNN to the task of recognition of handwritten digits. Since then, it has been used successfully used for various recognition and segmentation tasks. For instance, [110] uses them for automatic segmentation and detection of cells and nuclei in microscopic images. [111] applied the CNNs to recognize the traffic signs. [112] propose a CNN architecture for object class segmentation and apply it on the INRIA-Graz02 dataset. [113] have presented a fully parameterizable GPU implementation of CNN. [114] also presented a CNN based method which segments neuronal structures in electron microscope images. [115] applied convolutional neural networks for human action recognition in videos.

In the following discussion the bold symbols are used for matrices and vectors, and non-bold symbols are used for scalars. A single element of a matrix or vector will be denoted by a non-bold symbol, e.g. the (x, y) element of a matrix $\boldsymbol{A}$ will be written as $A(x, y)$.

Before going on to discuss CNN in detail, we give a brief overview of the general processing chain for a 2D CNN here with the help of an example CNN depicted in 5.3. The CNN shown is figure 5.3 has a particular architecture and has relevance in our work. However, here we use this as an example to give a general overview of CNNs.

CNN being a deep learning network, the features learnt have to be hierarchical in nature. This means that, as we move ahead in the network, the convolutional layer should learn features which are more abstract than the features learnt in the previous layers. Therefore, the output of each convolutional layer is sent through an non-linear activation function. In the discussion further L_C will denote convolutional layer, L_S will denote subsampling layer and L_F will denote fully-connected layer. 5.3 depicts an example of convolutional neural network with

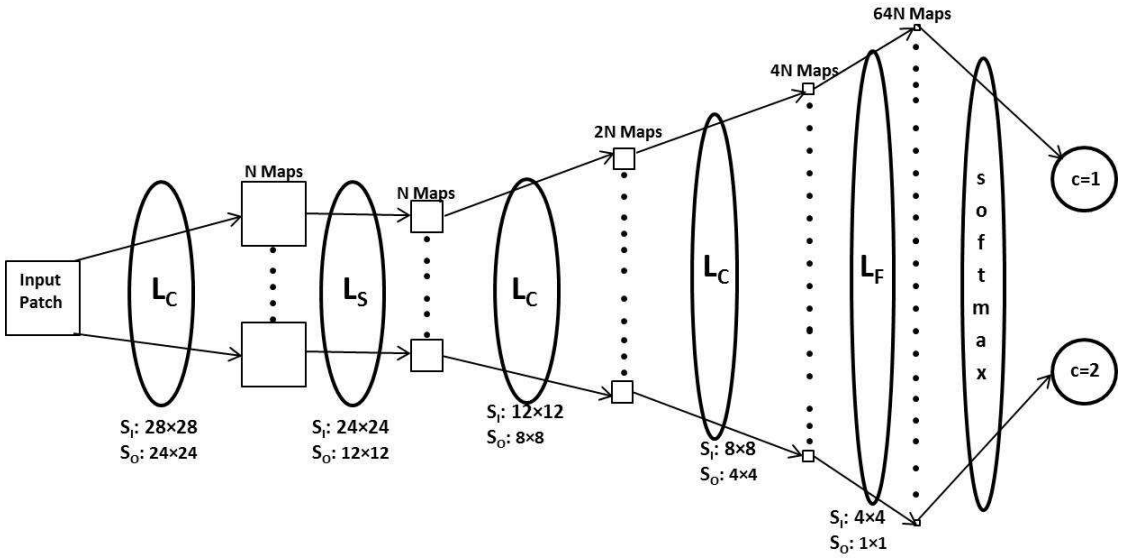


Figure 5.3: A 2D Convolutional neural network. L_C denotes convolutional layer, L_S denotes subsampling layer and L_F denotes fully-connected layer. Each pair of output and input maps of a convolutional layer has a 2D kernel linking them. Each output map of convolutional and subsampling layer has a bias parameter associated. The classifier involved is a softmax classifier which classifies the input patch into one of the two classes.

layer sequence $L_C \rightarrow L_S \rightarrow L_C \rightarrow L_C \rightarrow L_F$. Let an input patch of size 28×28 be the input to the CNN. The input patch is then convolved with N different kernels of size 5×5 to give N different output maps. These convolutions are performed without adding any zero padding to the input maps. Thus the sidelength of the N output maps of the first convolutional layer is $24(28 - 5 + 1)$. The output maps of each convolutional layer are sent through a non-linear activation function (usually a sigmoid) which is applied to each element of the maps. Apart from the convolutional kernels, the other learnable parameters for this layer are bias parameters which are unique for each output map. Thus the total number of learnable parameters for this layer are $1 * N * 5 * 5 + N = 26N$. Now the next layer is a subsampling layer (L_S). The role of subsampling layer is to reduce the number of parameters to save computations and avoid over-fitting. Our subsampling layer is a mean pooling layer. In the example shown in figure 5.3, the sub-sampling factor is 2, which means the input-maps of size 24×24 are reduced to 12×12. The number

of output maps and input maps are the same for a subsampling layer. Each input map is divided into 2×2 and average of these 2×2 blocks is saved as an element of 12×12 sized output map. The only parameters to be learnt for subsampling layer are the bias parameters. Thus the number of parameters for subsampling layer is N. As the next layer is a convolutional layer with $2N$ output maps, and N input maps, the number of kernels for this layer is $2N^2$, making the total number of kernel parameters equal to $(2N^2) * 5 * 5 = 50N^2$. Including the bias parameters, this layer has total number of parameters equal to $50N^2 + 2N$. The sidelength of the output map for this layer is $12 - 5 + 1 = 8$. In the example shown in figure 5.3 the next layer is also a convolutional layer, with $4N$ output maps. This layer has $5 * 5 * 8N^2 + 4N = 200N^2 + 4N$ parameters (including kernel and bias parameters) and outmap size 4×4. The output maps of this layer are vectorized and concatenated to form the output of the fully-connected layer. The number of output features for fully-connected layer is $(200N^2 + 4N) * 4 * 4 = (200N^2 + 4N) * 16$. The fully-connected layer has no parameters to learn. The output of fully-connected layer is given as input to the softmax classifiers. If the number of classes is 2, the number of softmax parameters is $2 * (8N^2 + 4N) * 16 = 32 * (8N^2 + 4N)$. All the parameters are learnt using gradient based optimization methods and the partial derivatives are calculated using backpropagation algorithm. As it is very difficult to cross-validate for all the possible design choices, the design choices for the overall architecture of a CNN are a combination of heuristics (e.g. [116]) and cross-validation. For the CNN in the figure we doubled the number of maps in each convolutional layer and thus we followed the trend $N \to N \to 2N \to 4N \to 64N$ for number of feature maps in each layer. The value of N was found through cross-validation.

5.3 Layers and Cost Function

Let $\boldsymbol{d}^{(l-1)^{(m)}}$ and $\boldsymbol{d}^{(l)^{(m)}}$ be the input and output for the lth layer respectively, corresponding to the mth training example. Here $\boldsymbol{d}^{(l-1)^{(m)}}$ is a combined notation for all the input maps of layer l corresponding to mth training input patch. Similarly $\boldsymbol{d}^{(l)^{(m)}}$ is a combined notation for all the output maps of layer l corresponding to the mth training input patch. The notation with the *superscript* (m) will refer to

the mth training example through-out this discussion. Let $\boldsymbol{d}^{(0)(m)}$ be the 2D input image patch and $\boldsymbol{d}^{(L)(m)}$ be the output of the last layer L for the same example. Let $S_I^{(l)} \times S_I^{(l)}$ and $S_O^{(l)} \times S_O^{(l)}$ be the size of the input and the output map, respectively, for layer l. Furthermore, let $N_I^{(l)}$ and $N_O^{(l)}$ be the number of input and output maps respectively for that layer. The input to lth layer is the output of $(l-1)$th layer, $N_I^{(l)} = N_O^{(l-1)}$ and $S_I^{(l)} = S_O^{(l-1)}$. We denote the jth output-feature-map of layer l as $\boldsymbol{d}_j^{(l)(m)}$. As the input-feature-maps of the lth layer are actually the output-feature-maps of the $(l-1)$th layer, $\boldsymbol{d}_i^{(l-1)(m)}$ is the ith input-feature-map of layer l.

5.3.1 Convolutional Layer

The output of a convolutional layer is computed as

$$\boldsymbol{d}_j{}^{(l)(m)} = f\Big(\sum_i \boldsymbol{d}_i{}^{(l-1)(m)} * \boldsymbol{w}_{ij}^{(l)} + b_j^{(l)} \mathbf{1}_{S_O^{(l)}} \Big) \ , \tag{5.1}$$

where $0 \leq i < N_I^{(l-1)}$, $0 \leq j < N_O^{(l-1)}$ and $*$ denotes the convolution without adding the zero padding, through-out our discussion. Thus if $\boldsymbol{A}$ has size $S_a \times S_a$ and $\boldsymbol{B}$ has size $S_b \times S_b$, then assuming $S_a \geq S_b$, the size of $\boldsymbol{A} * \boldsymbol{B}$ will always be $(S_a - S_b + 1) \times (S_a - S_b + 1)$. Let $\boldsymbol{C}$ be the output matrix obtained by convolving $\boldsymbol{A}$ with $\boldsymbol{B}$. Thus

$$\boldsymbol{C} = \boldsymbol{A} * \boldsymbol{B} \tag{5.2}$$

then

$$C(m,n) = \sum_{i=-\lfloor S_b/2 \rfloor}^{\lfloor S_b/2 \rfloor} \sum_{j=-\lfloor S_b/2 \rfloor}^{\lfloor S_b/2 \rfloor} A(m-i, n-j) B(i,j)$$
$$\text{where} \quad -\lceil \frac{S_a - S_b + 1}{2} \rceil \leq m, n \leq \lfloor \frac{S_a - S_b + 1}{2} \rfloor \tag{5.3}$$

The mapping f is a *nonlinear activation function*, often a sigmoid, which is applied to each component of its argument individually. The 4D tensor $\boldsymbol{w}^{(l)}$ is of size $S_W \times S_W \times N_I^{(l)} \times N_O^{(l)}$, and $\boldsymbol{w}_{ij}^{(l)}$ is the kernel linking ith input map to jth output map. The size of $\boldsymbol{w}_{ij}^{(l)}$ is $S_W \times S_W$, and we refer to S_W as the sidelength of the kernel. Finally, the scalar $b_j^{(l)}$ is the bias element for jth output-feature-map of

lth layer. $b_j^{(l)}$ is multiplied to every element of all-ones matrix $\mathbf{1}_{S_O^{(l)}}$ of size $S_O^{(l)} \times S_O^{(l)}$ before being added to $\sum_i \boldsymbol{d}_i^{(l-1)^{(m)}} * \boldsymbol{w}_{ij}^{(l)}$. The parameters for the convolutional layer are the kernel parameters and the bias parameters.

5.3.2 Subsampling Layer

We use *mean pooling* in the subsampling layer: the output of the element (x, y) of jth feature map of layer l is given as

$$d_j^{(l)^{(m)}}(x,y) = \frac{\sum_{q=0}^{s-1}\sum_{r=0}^{s-1} d_j^{(l-1)^{(m)}}(s \times x + q, s \times y + r)}{s^2} + b_j^{(l)} \ , \tag{5.4}$$

where s is the subsampling factor and $0 \leq x, y \leq S_I^{(l)} - 1$. A subsampling layer has the same number of output and input maps. The parameters for a subsampling layer are only bias parameters.

5.3.3 Fully-connected Layer

The final layer of a CNN is fully connected to the preceding layer. If L is the number of layers, we vectorize the output maps of layer $L-1$ and then concatenate them all together to obtain the output of the last layer. The number of output maps $N_O^{(L)}$ for this layer is $N_O^{(L-1)} \times \left(S_O^{(L-1)}\right)^2$, each corresponding to a single scalar. The output of the last layer is

$$d_j^{(L)^{(m)}}(0,0) = d_i^{(L-1)^{(m)}}(x,y) \ \text{ with } \ j = i \times (S_O^{(L-1)})^2 + y \times S_O^{(L-1)} + x \ . \tag{5.5}$$

5.3.4 Softmax Classifier

The output of the fully-connected layer L acts as input for the softmax classifier (i.e., logistic regression for a two class problem). Let $\boldsymbol{\theta}$ be the parameter matrix for softmax classifier. It has size $K \times N_O^{(L)}$, where K in the number of classes, each row being associated to one particular class. Let vectors ${\boldsymbol{\theta}_1}^T, {\boldsymbol{\theta}_2}^T \ldots \boldsymbol{\theta}_K^T$ be K rows of $\boldsymbol{\theta}$. Given the softmax matrix $\boldsymbol{\theta}$, the probability of mth training example belonging to class n is

$$p(n|\boldsymbol{d}^{(L)^{(m)}}; \boldsymbol{\theta}) = \frac{e^{{\boldsymbol{\theta}_n}^T \boldsymbol{d}^{(L)^{(m)}}}}{\sum_{c=1}^{K} e^{{\boldsymbol{\theta}_c}^T \boldsymbol{d}^{(L)^{(m)}}}} \ , \tag{5.6}$$

where $n \in \{1, 2, \ldots . K\}$ and $\boldsymbol{d}^{(L)^{(m)}}$ is the output of the last layer of the CNN when presented the mth training example.

Cost Function As $\boldsymbol{d}^{(L)^{(m)}}$ is dependent on $\boldsymbol{d}^{(0)^{(m)}}$, and the kernel and bias parameters of the CNN, $p(n|\boldsymbol{d}^{(L)^{(m)}}; \boldsymbol{\theta})$ can be seen as the probability of mth training example belonging to class n given the CNN parameters, input patch, and the softmax parameters. Let $\boldsymbol{\Omega}$ be a set of all the parameters comprising the kernel and the bias parameters from all the layers of the CNN as well as the softmax parameters, $p(n|\boldsymbol{d}^{(L)^{(m)}}; \boldsymbol{\theta})$ can also be written as $p(n|\boldsymbol{d}^{(0)^{(m)}}; \boldsymbol{\Omega})$. Let $\boldsymbol{t}$ be a vector which stores the true labels of all the training data points and $t^{(m)}$ be the true label of the mth training example. The parameters are identified using a maximum likelihood approach. The cost function associated with the mth training example is

$$Q^{(m)} = -\ln(p(t^{(m)}|\boldsymbol{d}^{(0)^{(m)}}; \boldsymbol{\Omega})) \tag{5.7}$$

For all M training examples we have accordingly

$$Q = \frac{1}{M} \sum_{m=1}^{M} Q^{(m)} \tag{5.8}$$

We use *weight decay*, which penalizes too large values of the softmax-parameters, to regularize the classification. The final cost function is given as

$$Q_\lambda = Q + \frac{\lambda}{2} \sum_{p=1}^{K} \sum_{q=1}^{S_O^{(L)}} \theta_{pq}^2. \tag{5.9}$$

The optimum λ is chosen usually by cross-validation. Role of λ can be compared to the regularization parameter C in SVMs. However, increasing the value of C leads us to overfitting in SVMs, while decreasing the value of λ leads us to overfitting.

The cost function is minimized by gradient-based optimization and the gradients are computed using backpropagation. LBFGS [1] has proven to be well suited

[1]LBFGS: Limited-memory Broyden–Fletcher–Goldfarb–Shanno

for the optimization [117].

5.4 Gradient w.r.t. Softmax Parameters

The gradient with respect to softmax parameters can be calculated as following. Let Q_λ be the cost function given in equation (5.9), the gradient with respect to $\boldsymbol{\theta}_c$ is given as

$$\frac{\partial Q_\lambda}{\partial \boldsymbol{\theta}_c} = -\frac{1}{M}\sum_{m=1}^{M}\frac{1}{p(t^{(m)}|\boldsymbol{\theta}_c;\boldsymbol{d}^{(L)^{(m)}})}\frac{\partial p(t^{(m)}|\boldsymbol{\theta}_c;\boldsymbol{d}^{(L)^{(m)}})}{\partial \boldsymbol{\theta}_c} + \lambda\boldsymbol{\theta}_c \quad (5.10)$$

The above equation needs gradient $\frac{\partial p(t^{(m)}|\boldsymbol{\theta}_c;\boldsymbol{d}^{(L)^{(m)}})}{\partial \boldsymbol{\theta}_c}$ to be calculated which can be easily done using the following equation

$$\frac{\partial p(t^{(m)}|\boldsymbol{\theta}_c;\boldsymbol{d}^{(L)^{(m)}})}{\partial \boldsymbol{\theta}_c} = p(t^{(m)}|\boldsymbol{\theta}_c;\boldsymbol{d}^{(L)^{(m)}})(\mathbb{1}(t^{(m)},c) - p(c|\boldsymbol{\theta}_c;\boldsymbol{d}^{(L)^{(m)}}))\boldsymbol{d}^{(L)^{(m)}} \quad (5.11)$$

where $\mathbb{1}(p,q)$ be a function returning one if $p=q$ and zero otherwise.

5.5 Backpropogation for Convolutional Neural Networks

The partial derivatives for the gradient based optimization of the CNN cost function are calculated using backpropagation [118]. The basic idea behind the backpropagation algorithm is to calculate the activations of the each node (each element of the feature maps), compute the error in the output and then find out how much a particular node contributes in the error. These error contributions are called sensitivities which are then used to calculate the partial derivatives.

In the following, first we will discuss the computation of the sensitivities and then the calculation of the partial derivatives.

5.5.1 Sensitivity Calculation

The sensitivity map of each layer is dependent on the layer next to it in the CNN configuration. We discuss each case one by one. The following discussion on

sensitivity calculation is done assuming that the example in consideration is mth example.

Layer Followed by the Softmax Classifier, i.e. Current Layer Being the Last Layer The last layer which is fully-connected layer, is followed by the softmax classifier. The sensitivity map of the last layer is dependent on the softmax parameters, posterior probability and the input label of the example. Let $\boldsymbol{D}^{(m)}$ be a vector given as

$$\boldsymbol{D}^{(m)} = \begin{bmatrix} \mathbb{1}(t^{(m)}, 1) - p(t^{(m)}|\boldsymbol{\theta}_1; \boldsymbol{d}^{(L)^{(m)}}) \\ \mathbb{1}(t^{(m)}, 2) - p(t^{(m)}|\boldsymbol{\theta}_2; \boldsymbol{d}^{(L)^{(m)}}) \\ \vdots \\ \mathbb{1}(t^{(m)}, c) - p(t^{(m)}|\boldsymbol{\theta}_c; \boldsymbol{d}^{(L)^{(m)}}) \\ \vdots \\ \mathbb{1}(t^{(m)}, K) - p(t^{(m)}|\boldsymbol{\theta}_K; \boldsymbol{d}^{(L)^{(m)}}) \end{bmatrix} \tag{5.12}$$

Now, the sensitivity map of the last layer is given as

$$\boldsymbol{\delta}^{(L)^{(m)}} = -\boldsymbol{\theta}^T \boldsymbol{D}^{(m)} \tag{5.13}$$

Next Layer being the Last Layer, i.e. a Fully-connected Layer Last layer is just the vectorized and concatenated version of the previous layer. Thus, if $(L-1)$th layer is convolutional layer, the (x, y) element of the sensitivity map is given as

$$\delta_i^{(L-1)^{(m)}}(x, y) = f'(d^{(L-1)^{(m)}}(x, y))\delta_j^{(L)^{(m)}}(0, 0) \text{ with } i = \frac{j - x - y \times S_O^{(L-1)}}{(S_O^{(L-1)})^2} \tag{5.14}$$

If $(L-1)$th layer is subsampling layer, the (x, y) element of the sensitivity map is given as

$$\delta_i^{(L-1)^{(m)}}(x, y) = \delta_j^{(L)^{(m)}}(0, 0) \text{ with } i = \frac{j - x - y \times S_O^{(L-1)}}{(S_O^{(L-1)})^2} \tag{5.15}$$

The term $f'({d^{(l)}}^{(m)}(x,y))$ appears in first case ((5.14)) because the final activations for the convolutional layer are obtained by applying a non-linear function as given in equation (5.1).

Next Layer being a Convolutional Layer Each element of a feature map of lth layer contributes to a patch (same as the size of kernel) of elements in each feature map of the subsequent $((l+1)\text{th})$ convolutional layer. The contribution weights corresponding to a patch are the weights of the kernel The sensitivity maps of the convolutional layer $((l+1)\text{th})$ are zero padded before computing the sensitivities of the the lth layer.

Let ${\boldsymbol{\delta}_{j_z}}^{(l+1)}$ be the zero padded sensitivity map of size $2S_W - 2 + S_O^{(l+1)}$ for the jth feature map of the $(l+1)$th layer.

$$\left.\begin{aligned} {\delta_{j_z}}^{(l+1)^{(m)}}(u,v) &= 0 \quad \text{if} \quad S_W - 1 \geq u,v \ \text{ or } \ u,v \geq S_O^{(l+1)} + S_W - 1 \\ {\delta_{j_z}}^{(l+1)^{(m)}}(u,v) &= {\delta_j}^{(l+1)^{(m)}}(u,v) \quad \text{otherwise} \end{aligned}\right\} \tag{5.16}$$

where $0 \leq u,v \leq 2S_W - 2 + S_O^{(l+1)}$

Let ${\boldsymbol{\chi}_{j_{p,q}}}^{(l+1)^{(m)}}$ be the patch of size $S_O^{(l)} \times S_O^{(l)}$ extracted from ${\boldsymbol{\delta}_{j_z}}^{(l+1)^{(m)}}$ centered at (p,q).

Now, if layer l is a convolutional layer, ${\boldsymbol{\delta}_i}^{(l)^{(m)}}$ can be written as

$${\boldsymbol{\delta}_i}^{(l)^{(m)}} = f'({\boldsymbol{d}_i}^{(l)^{(m)}}) \circ \sum_{j=0}^{N_O^{(l+1)}-1} \sum_{u=0}^{S_W-1} \sum_{v=0}^{S_W-1} (({w_{ij}}^{(l+1)}(u,v)){\boldsymbol{\chi}_{j_{p,q}}}^{(l+1)^{(m)}}) \tag{5.17}$$

with $p = u + \lfloor S_O^{(l)}/2 \rfloor$ and $q = v + \lfloor S_O^{(l)}/2 \rfloor$ and $\circ$ denotes element wise multiplication of the matrices. In the above equation the (u,v) element of ${\boldsymbol{w}_{ij}}^{(l+1)}$ is multiplied to each element of ${\boldsymbol{\chi}_{j_{p,q}}}^{(l+1)^{(m)}}$. Term $\sum_{u=0}^{S_W-1}\sum_{v=0}^{S_W-1}(({w_{ij}}^{(l+1)}(u,v)){\boldsymbol{\chi}_{j_{p,q}}}^{(l+1)^{(m)}})$ in the above equation is nothing but the correlation between the zero padded sensitivity map ${\boldsymbol{\delta}_{j_z}}^{(l+1)^{(m)}}$ and the kernel, which is same as convolution between ${\boldsymbol{\delta}_{j_z}}^{(l+1)^{(m)}}$ and 180° rotated kernel. The (x,y) element of the rotated kernel ${\boldsymbol{w}_{ij_r}}^{(l+1)}$ can be written as

$${w_{ij_r}}^{(l+1)}(x,y) = {w_{ij}}^{(l+1)}(S_W - x - 1, S_W - y - 1) \tag{5.18}$$

where $0 \leq x, y \leq S_W - 1$

Thus (5.17) can be re-written as

$$\boldsymbol{\delta}_i{}^{(l)(m)} = f'(\boldsymbol{d}_i{}^{(l)(m)}) \circ \sum_{j=0}^{N_O^{(l+1)}-1} (\boldsymbol{w}_{ij_r}{}^{(l+1)} * \boldsymbol{\delta}_{j_z}{}^{(l+1)(m)}) \tag{5.19}$$

As mentioned earlier f is non linear function, usually a sigmoid. For a sigmoid function, the derivative can be easily calculated as

$$f'(z) = f(z)(1 - f(z)) \tag{5.20}$$

If l is a subsampling layer $\boldsymbol{\delta}_i{}^{(l)(m)}$ can be written as

$$\boldsymbol{\delta}_i{}^{(l)(m)} = \sum_{j=1}^{N_O^{(l+1)}} (\boldsymbol{w}_{ij_r}{}^{(l+1)} * \boldsymbol{\delta}_{j_z}{}^{(l+1)(m)}) \tag{5.21}$$

Next Layer being a Subsampling Layer Let $\boldsymbol{\delta}_i^{(l)(m)}$ be the sensitivity map of the lth layer which we need to calculate assuming that we know the sensitivity map $\boldsymbol{\delta}_i^{(l+1)(m)}$ of the $(l+1)$th layer, which is a subsampling layer. We know that being a mean pooling layer, each element of the subsampling layer's feature maps is an average of a block of $s \times s$ elements of a feature map of the previous layer. Thus each element of an $s \times s$ block from ith feature map of the lth layer contributes equally to the sensitivity of the corresponding element from the ith feature map in the subsampling layer ($(l+1)$th layer). The following equations ((5.22), (5.23)) give the (x, y) element of the sensitivity map of the lth layer, given $(l+1)$th layer is a subsampling layer and s is the subsampling factor.

If the lth layer is a convolutional layer then the final sensitivity map is given as

$$\delta_i^{(l)(m)}(x, y) = f'(d^{(l)(m)}(x, y)) \circ \delta_i^{(l+1)(m)}(\lfloor x/s \rfloor, \lfloor y/s \rfloor)/s^2 \tag{5.22}$$

where $\circ$ denotes element-wise multiplication.

Else, if the lth layer is a subsampling layer then the final sensitivity map is given as

$$\delta_i^{(l)(m)}(x, y) = \delta_i^{(l+1)(m)}(\lfloor x/s \rfloor, \lfloor y/s \rfloor)/s^2 \tag{5.23}$$

5.5.2 Gradient Calculation

We will discuss the two cases here. First the case when the current layer is a convolutional layer.

Current Layer being Convolutional Layer The bias gradient can be simply obtained by summing over all the entries of the sensitivity map of that feature map. Thus

$$\frac{\partial Q}{\partial b_j^{(l)}} = \frac{1}{M} \sum_{m=1}^{M} \sum_{x=0}^{S_O^{(l)}-1} \sum_{y=0}^{S_O^{(l)}-1} {\boldsymbol{\delta}_j}^{(l)^{(m)}}(x, y) \tag{5.24}$$

In a neural network where we have a unique weight for each connection between an input and output node, the gradient with respect to that weight is given by the product between the sensitivity of corresponding output node and activation of the corresponding input node. However, in case of convolutional layer, each weight is shared by many output-input node combination. Therefore, here we need to sum the gradients over all the connections which connect through this weight as done in following equation which gives the gradient with respect to the rotated kernel. Rotated because while performing convolution, the kernel is rotated before performing elementwise multiplication.

$$\frac{\partial Q}{\partial \boldsymbol{w_r}_{ij}^{(l)}} = \frac{1}{M} \sum_{m=1}^{M} \sum_{u=0}^{S_O^{(l)}-1} \sum_{v=0}^{S_O^{(l)}-1} ((\delta_j{}^{(l)^{(m)}}(u, v)) {\boldsymbol{\zeta}_{ip,q}}^{(l-1)^{(m)}}) \tag{5.25}$$

Here ${\boldsymbol{\zeta}_{ip,q}}^{(l-1)^{(m)}}$ is the patch of size $S_W \times S_W$ extracted from ${\boldsymbol{d}_i}^{(l-1)^{(m)}}$ centered at (p, q) with $p = u + \lfloor S_W/2 \rfloor$ and $q = v + \lfloor S_W/2 \rfloor$. Element (u, v), of ${\boldsymbol{\delta}_j}^{(l)^{(m)}}$ is multiplied with each element of ${\boldsymbol{\zeta}_{ip,q}}^{(l-1)^{(m)}}$. The above equation can be written as correlation between the input ${\boldsymbol{d}_i}^{(l-1)^{(m)}}$ and sensitivity map ${\boldsymbol{\delta}_j}^{(l)^{(m)}}$, which is same as convolution between the input ${\boldsymbol{d}_i}^{(l-1)^{(m)}}$ and the 180° rotated sensitivity map ${\boldsymbol{\delta}_j}^{(l)^{(m)}}$. Let ${\boldsymbol{\delta}_{j_r}}^{(l)^{(m)}}$ be the 180° rotated sensitivity map, the gradient with respect to 180° rotated kernel $\frac{\partial Q}{\partial \boldsymbol{w_r}_{ij}^{(l)}}$ can be given as

$$\frac{\partial Q}{\partial \boldsymbol{w_r}_{\boldsymbol{ij}}^{(l)}} = \frac{1}{M} \sum_{m=1}^{M} {\boldsymbol{\delta}_{j_r}}^{(l)(m)} * {\boldsymbol{d}_i}^{(l-1)(m)} \tag{5.26}$$

Finally, the gradient $\frac{\partial Q}{\partial \boldsymbol{w_r}_{\boldsymbol{ij}}^{(l)}}$ can be rotated 180° to get the gradient $\frac{\partial Q}{\partial \boldsymbol{w}_{\boldsymbol{ij}}^{(l)}}$

Current Layer being Subsampling Layer The subsampling layer has only bias parameters to learn. The bias gradient can be obtained in the same way as convolutional layer (see equation (5.24)).

The next chapter introduces our triplanar convolutional neural network for classifying voxels in 3D images and shows its application on segmenting tibial cartilage in knee MRIs.

Chapter 6

Deep Feature Learning for Knee Cartilage Segmentation Using a Triplanar Convolutional Neural Network

Abstract

Segmentation of anatomical structures in medical images is often based on a voxel/pixel classification approach. Deep learning systems, such as convolutional neural networks (CNNs), can infer a hierarchical representation of images that fosters categorization. We propose a novel system for voxel classification integrating three 2D CNNs, which have a one-to-one association with the xy, yz and zx planes of 3D image, respectively. We applied our method to the segmentation of tibial cartilage in low field knee MRI scans and tested it on 114 unseen scans. Although our method uses only 2D features at a single scale, it performs better than a state-of-the-art method

This chapter is based on "Deep Feature Learning for Knee Cartilage Segmentation Using a Triplanar Convolutional Neural Network" by Adhish Prasoon, Kersten Petersen, Christian Igel, Francois Lauze, Erik Dam, and Mads Nielsen, published in the proceedings of 16th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 2013). [56]

using 3D multi-scale features. In the latter approach, the features and the classifier have been carefully adapted to the problem at hand. That we were able to get better results by a deep learning architecture that autonomously learns the features from the images is the main insight of this study.

6.1 Introduction

Convolutional neural networks (CNNs, [47]) are deep learning architectures and have recently been employed successfully for 2D image segmentation tasks [119]. Here we use them for voxel classification in 3D images. There is a generic extension of CNNs from 2D to 3D images [120]. However, these truly 3D CNNs have large memory and training time requirements, retarding their application for time-constrained large-scale medical imaging tasks. In this paper we propose a classifier based on a system of *triplanar 2D CNNs* which classifies voxels from 3D images with high accuracy and is easy to train.

Deterioration of cartilage implies Osteoarthristis, which is one of the major reasons for work disability [73]. Cartilage segmentation in knee MRI scans is the method of choice for quantitative analysis of cartilage deterioration. We applied our method for segmenting tibial articular cartilage from knee MRI and compared the performance with a state-of-the-art method [71]. Cartilage segmentation in knee MRI scans is often performed by the radiologists in a slice-by-slice manner and is a time consuming and tedious job. Furthermore, the inter and intra observer variability is found to be rather high. Thus, for large studies, automated segmentation is desirable. Therefore, the automatic and semi-automatic segmentation of knee cartilage in MRI scans is an active field of research. These methods address the task either slice-by-slice or directly perform 3D segmentation [78, 79, 89, 95]. In this thesis, we compare our methods with Folkesson et al. [71]. However, we have also discussed some other recent and state-of-the-art [91, 92, 93, 94, 95] works on knee cartilage segmentation (please see section 3.1.2).

A State-of-the-art method by Folkesson et al. Folkesson et al. [71] developed a highly efficient voxel classification based segmentation method which can be considered as one the state-of-the-art in fully automatic cartilage segmentation from MRI scans. It served as the reference in our study. The approach is based on k nearest neighbor classification (kNN) with $k = 100$ and a predefined 178-dimensional feature vector. The training of their classifier involves reducing the number of features using a sequential floating forward feature selection algorithm and also selecting a posterior threshold that optimizes the segmentation results. The features calculated using 3D filters are: the intensity, the three-jet, the position, the eigenvectors and eigenvalues of the 3D Hessian matrix, as well as the eigenvector and eigenvalues of the 3D structure tensor and the third order tensor in the gradient direction. All features except position are calculated at 3 different scales.

6.2 Method

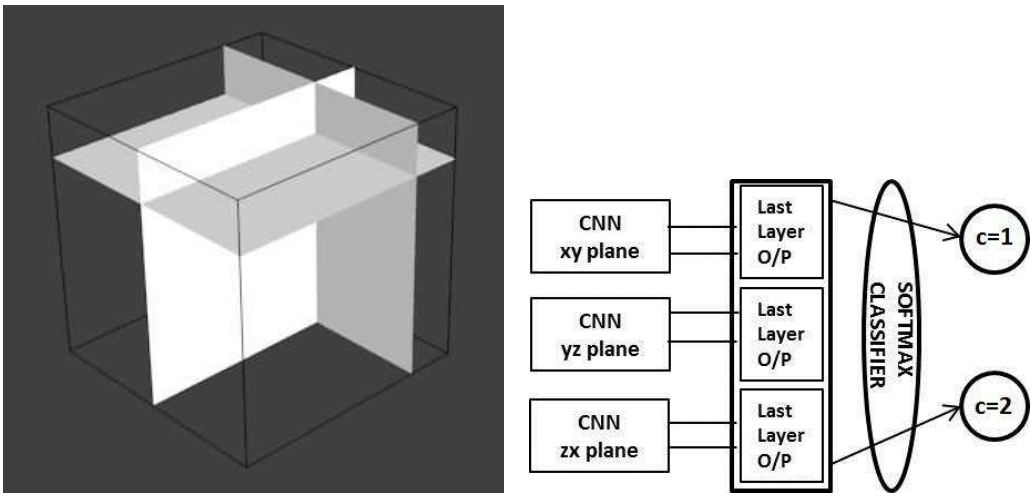
Convolutional neural networks were introduced by LeCun et al. [47], who applied it for handwritten digit recognition. Since then, CNNs have been used successfully for many recognition and segmentation tasks [119, 110, 111, 113, 121, 115].

6.2.1 Convolutional Neural Networks.

Please see chapter 5 for a detailed discussion on Convolutional Neural Networks

6.2.2 Triplanar Convolutional Neural Network.

For mth example voxel, we take the three planes $\boldsymbol{P}_{xy}^{(m)}, \boldsymbol{P}_{yz}^{(m)}, \boldsymbol{P}_{zx}^{(m)}$ that pass through the voxel and are parallel to the xy, yz and zx plane, respectively, of the 3D image (see Fig. 6.1a). A 2D patch centered around that voxel is extracted from each of the three planes. Let $\boldsymbol{d}_{xy}{}^{(0)(m)}, \boldsymbol{d}_{yz}{}^{(0)(m)}, \boldsymbol{d}_{zx}{}^{(0)(m)}$ be those three patches. We have one CNN for each of the three patches. The 3 CNNs are not connected with each other except for the output of the last layer. Let $\boldsymbol{d}_{xy}{}^{(L)(m)}, \boldsymbol{d}_{yz}{}^{(L)(m)}$ and $\boldsymbol{d}_{zx}{}^{(L)(m)}$ be the outputs of the last layers of the three



(a) **(b)**

Figure 6.1: The three image planes giving rise to our triplanar convolutional neural network (CNN) architecture. One patch centered in the voxel is extracted from each of the planes. The three CNNs are fused in the final layer.

CNNs and $N_{O_{xy}}^{(L)}, N_{O_{yz}}^{(L)}$ and $N_{O_{zx}}^{(L)}$ be the number of output-maps of the last layers of the three CNNs. The last layers' output-maps for all 3 CNNs are scalars. Thus ${\boldsymbol{d}_{xy}}^{(L)^{(m)}}, {\boldsymbol{d}_{yz}}^{(L)^{(m)}}$ and ${\boldsymbol{d}_{zx}}^{(L)^{(m)}}$ are vectors of length $N_{O_{xy}}^{(L)}$, $N_{O_{yz}}^{(L)}$, and $N_{O_{zx}}^{(L)}$, respectively. ${\boldsymbol{d}_{xy}}^{(L)^{(m)}}, {\boldsymbol{d}_{yz}}^{(L)^{(m)}}$ and ${\boldsymbol{d}_{zx}}^{(L)^{(m)}}$ are concatenated to obtain a joint output $\boldsymbol{d}_T^{(L)^{(m)}}$, which is fed into the softmax classifier. The softmax parameter matrix $\boldsymbol{\theta}_T$ in this case is of size $K \times N_{OT}$, where $N_{OT} = N_{O_{xy}}^{(L)} + N_{O_{yz}}^{(L)} + N_{O_{zx}}^{(L)}$. Let $\boldsymbol{\Omega}_T$ be the set of all the parameters collecting the kernel and bias parameters from all the layers of the three CNNs as well as the softmax parameters. The cost associated with the mth training example in the triplanar CNN is

$$ {Q_T}^{(m)} = -\ln(p(t^{(m)}|{\boldsymbol{d}_{xy}}^{(0)^{(m)}}; {\boldsymbol{d}_{yz}}^{(0)^{(m)}}; {\boldsymbol{d}_{zx}}^{(0)^{(m)}}; \boldsymbol{\Omega}_T)) \tag{6.1} $$

and the overall cost function reads

$$ Q_{T\lambda} = -\frac{1}{M}\sum_{m=1}^{M} \ln(p(t^{(m)}|{\boldsymbol{d}_{xy}}^{(0)^{(m)}}; {\boldsymbol{d}_{yz}}^{(0)^{(m)}}; {\boldsymbol{d}_{zx}}^{(0)^{(m)}}; \boldsymbol{\Omega}_T)) + \frac{\lambda}{2}\sum_{p=1}^{K}\sum_{q=1}^{N_{OT}} \theta_{T_{pq}}^2 \ , \tag{6.2} $$

where $\theta_{T_{pq}}$ is the (p, q) element of the matrix $\boldsymbol{\theta}_T$ and λ controls the strength of the weight decay. As

$$p(t^{(m)}|\boldsymbol{d}_{xy}{}^{(L)^{(m)}}; \boldsymbol{d}_{yz}{}^{(L)^{(m)}}; \boldsymbol{d}_{zx}{}^{(L)^{(m)}}; \boldsymbol{\theta}_T) = p(t^{(m)}|\boldsymbol{d}_{xy}{}^{(0)^{(m)}}; \boldsymbol{d}_{yz}{}^{(0)^{(m)}}; \boldsymbol{d}_{zx}{}^{(0)^{(m)}}; \boldsymbol{\Omega}_T), \tag{6.3}$$

the equation (6.2) can also be written as

$$Q_{T\lambda} = -\frac{1}{M}\sum_{m=1}^{M} \ln(p(t^{(m)}|\boldsymbol{d}_{xy}{}^{(L)^{(m)}}; \boldsymbol{d}_{yz}{}^{(L)^{(m)}}; \boldsymbol{d}_{zx}{}^{(L)^{(m)}}; \boldsymbol{\theta}_T)) + \frac{\lambda}{2}\sum_{p=1}^{K}\sum_{q=1}^{N_{OT}} \theta_{T_{pq}}^2 \ , \tag{6.4}$$

Assuming $\boldsymbol{d}_T{}^{(L)^{(m)}}$ be the vector obtained by concatenating $\boldsymbol{d}_{xy}{}^{(L)^{(m)}}$, $\boldsymbol{d}_{yz}{}^{(L)^{(m)}}$ and $\boldsymbol{d}_{zx}{}^{(L)^{(m)}}$, we can write the above equation as

$$Q_{T\lambda} = -\frac{1}{M}\sum_{m=1}^{M} \ln(p(t^{(m)}|\boldsymbol{d}_T{}^{(L)^{(m)}}; \boldsymbol{\theta}_T)) + \frac{\lambda}{2}\sum_{p=1}^{K}\sum_{q=1}^{N_{OT}} \theta_{T_{pq}}^2 \tag{6.5}$$

The gradient of the cost function with respect to the softmax parameters corresponding to class c can be given as

$$\frac{\partial Q_{T\lambda}}{\partial \boldsymbol{\theta}_{Tc}} = -\frac{1}{M}\sum_{m=1}^{M}\left[\frac{1}{p(t^{(m)}|\boldsymbol{\theta}_{Tc}; \boldsymbol{d}_T{}^{(L)^{(m)}})}\frac{\partial p(t^{(m)}|\boldsymbol{\theta}_{Tc}; \boldsymbol{d}_T{}^{(L)^{(m)}})}{\partial \boldsymbol{\theta}_{Tc}}\right] + \lambda\boldsymbol{\theta}_{Tc} \tag{6.6}$$

where $\boldsymbol{\theta}_{Tc}$ is the cth row of the softmax parameter matrix $\boldsymbol{\theta}_T$ of triplanar CNN. Gradient $\frac{\partial p(t^{(m)}|\boldsymbol{\theta}_{Tc}, \boldsymbol{d}^{(L)^{(m)}})}{\partial \boldsymbol{\theta}_{Tc}}$ can be easily calculated using

$$\frac{\partial p(t^{(m)}|\boldsymbol{\theta}_{Tc}; \boldsymbol{d}_T{}^{(L)^{(m)}})}{\partial \boldsymbol{\theta}_{Tc}} = p(t^{(m)}|\boldsymbol{\theta}_{Tc}; \boldsymbol{d}_T{}^{(L)^{(m)}})\boldsymbol{d}_T{}^{(L)^{(m)}}\Big[\mathbb{1}(t^{(m)}, c) - p(t^{(m)}|\boldsymbol{\theta}_{Tc}; \boldsymbol{d}_T{}^{(L)^{(m)}})\Big] \tag{6.7}$$

As mentioned earlier, softmax parameter matrix for triplanar CNN is a $K \times N_{OT}$, where $N_{OT} = N_{O_{xy}}^{(L)} + N_{O_{yz}}^{(L)} + N_{O_{zx}}^{(L)}$. Let $\boldsymbol{\theta}_{xy}$ be a matrix constructed by extracting first $N_{O_{xy}}^{(L)}$ columns of $\boldsymbol{\theta}_T$ and $\boldsymbol{\theta}_{yz}$ be a matrix constructed by extracting $(N_{O_{xy}}^{(L)} + 1)$th to $(N_{O_{xy}}^{(L)} + N_{O_{yz}}^{(L)})$th columns. Similarly, let $\boldsymbol{\theta}_{zx}$ be the matrix constructed by extracting $(N_{O_{xy}}^{(L)} + N_{O_{yz}}^{(L)} + 1)$th to $N_{O_T}^{(L)}$th columns of $\boldsymbol{\theta}_T$.

Following equations can be used to calculate the sensitivity map of the fully connected layers of the three CNNs corresponding to mth example

$$\boldsymbol{\delta}_{xy}{}^{(L)(m)} = -\boldsymbol{\theta}_{xy}{}^{T}\boldsymbol{D}^{(m)} \tag{6.8}$$

$$\boldsymbol{\delta}_{yz}{}^{(L)(m)} = -\boldsymbol{\theta}_{yz}{}^{T}\boldsymbol{D}^{(m)} \tag{6.9}$$

$$\boldsymbol{\delta}_{zx}{}^{(L)(m)} = -\boldsymbol{\theta}_{zx}{}^{T}\boldsymbol{D}^{(m)} \tag{6.10}$$

Vector $\boldsymbol{D}^{(m)}$ in this case is given as

$$\boldsymbol{D}^{(m)} = \begin{bmatrix} \mathbb{1}(t^{(m)}, 1) - p(t^{(m)}|\boldsymbol{\theta}_{T1}; \boldsymbol{d}_T{}^{(L)(m)}) \\ \mathbb{1}(t^{(m)}, 2) - p(t^{(m)}|\boldsymbol{\theta}_{T2}; \boldsymbol{d}_T{}^{(L)(m)}) \\ \vdots \\ \mathbb{1}(t^{(m)}, c) - p(t^{(m)}|\boldsymbol{\theta}_{Tc}; \boldsymbol{d}_T{}^{(L)(m)}) \\ \vdots \\ \mathbb{1}(t^{(m)}, K) - p(t^{(m)}|\boldsymbol{\theta}_{TK}; \boldsymbol{d}_T{}^{(L)(m)}) \end{bmatrix} \tag{6.11}$$

As the three CNNs are only connected with each other in the last layer, we can calculate the sensitivites maps and partial derivatives for three CNNs independently following the discussion in 5.5 using $\boldsymbol{\delta_{xy}}^{(L)(m)}, \boldsymbol{\delta_{yz}}^{(L)(m)}, \boldsymbol{\delta_{zx}}^{(L)(m)}$.

6.3 Application to Cartilage Segmentation in MRI Scans

In our cartilage segmentation experiments, we minimize $Q_{\lambda T}$ using LBFGS using mini-batches. For each mini-batch, we selected 2000 examples randomly from 120000 training examples and run 20 iterations of LBFGS. For each of the 3 CNNs we used exactly the same configuration: the sequence of layers as well as the number and size of feature maps in corresponding layers were the same. Let $\mathrm{L_C}$ denote a convolutional layer, $\mathrm{L_S}$ a subsampling layer, and $\mathrm{L_F}$ the final output layer. We used the sequence $\mathrm{L_C} \rightarrow \mathrm{L_S} \rightarrow \mathrm{L_C} \rightarrow \mathrm{L_C} \rightarrow \mathrm{L_F}$ for each CNN. The single subsampling layer performed mean pooling with subsampling factor 2 (see equation (5.4)). Instead of a second subsampling layer after the fourth-layer (a convolution layer), we have another convolution layer as the fifth layer. Having more than one subsampling layers can lead to over-summarizing of the features,

which did not seem to be a good approach for our application because of the thin structure of the cartilage. For each CNN, we fixed the size of input patches to 28×28 and kernel size to 5×5. Let N be the number output feature maps of the first convolutional layer. The numbers of output-feature-maps are $N \to N \to 2N \to 4N \to 64N$ for the whole sequence of layers. Fig. 5.3 depicts such a CNN. Combining three of these CNNs leads to the triplanar CNN for 3D images shown in Fig. 6.1b, where the output of the final CNN layers are concatenated before being fed into the classifier. We selected N using cross-validation (splitting the training dataset) from $\{12, 16, 20, 24, 28, 32\}$. We also selected the weight decay parameter λ using cross-validation from $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. As a result, $N = 28$ and $\lambda = 10^{-2}$ were selected. The heuristic design choices were inspired by Simard et al.'s article about best practices for CNNs for document analysis [116] and our previous experience in CNN design. A posterior threshold also chosen to maximize the segmentation performance.

The patient's knee is being scanned in standardized position in a specialized extremity MRI scanner. Still, there will be small variations from scan to scan between the image planes and corresponding body planes due to anatomical and placement differences. This variability is reflected by the training data. Thus, the triplanar CNN is trained to become invariant under these variations.

Training Data We use the same 25 training scans as in [71], where features from on average 84000 voxels from each of the scans are extracted. These 84000 voxels include all the cartilage voxels in a scan and sampled background voxels. As it is more difficult to classify the background voxels near the cartilage, they are sampled very densely close to the cartilage and rarely far from it, with sampling probability varying linearly. For generating our training data we extracted triplanar patches from 4800 randomly selected voxels from the pool of 84000 voxels used by [71] for each of the 25 scans. That is, while in [71], 2100000 voxels were used to generate the training data, we just considered 120000 voxels for limiting the computational cost and found this number to be sufficient. We applied ZCA whitening to preprocess our patches, which is very similar to PCA whitening, however, it also involves a

rotation back to the input data basis.

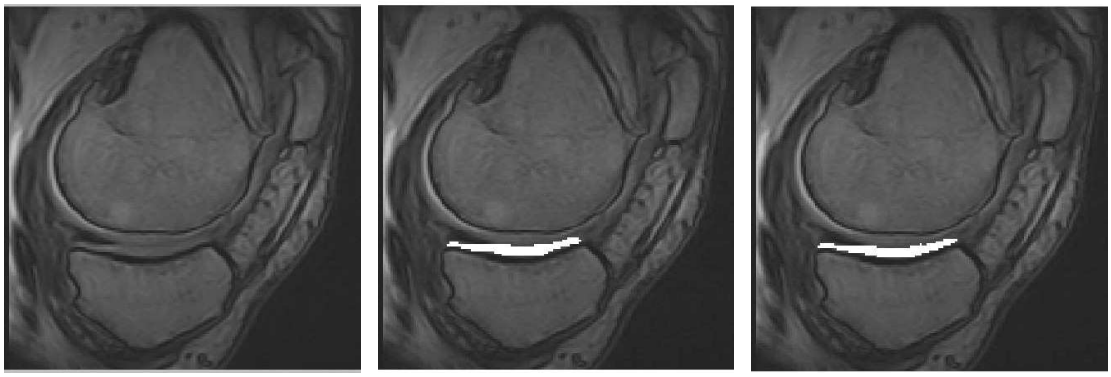
6.4 Evaluation and Results

We evaluated our method on 114 unseen knee MRIs, each having approximately 2 million voxels to classify. After classifying the voxels, the largest connected component was chosen as the final segmentation. We compared our method to the state-of-the-art method [71] which was also evaluated on the same test set of 114 scans. The evaluation was based on dice similarity coefficient(DSC) where $\mathrm{DSC}(A,B) = \frac{2(|A \cap B|)}{|A|+|B|}$ and A, B are manual and automatic segmentation. Table 6.1 shows results obtained by the triplanar-CNN and the kNN based method. Our method achieves better DSC, sensitivity, specificity and accuracy than [71]. The difference in DSC is statistically significant (Wilcoxon rank-sum test, $p < 0.05$). Fig. 6.2 shows a knee MRI slice, its segmentation by a radiologist, and the segmentation by our method.

Our method used just 120000 training voxels whereas in [71] 2.1 million training voxels are used. We also evaluated the performance of the method proposed in [71] using 120000 training voxels and observed a loss in performance. The average DSC, accuracy, sensitivity and specificity fell to 0.7952 ($\pm$9.10%), 99.91% ($\pm$3.18%) ,78.58% ($\pm$11.61%) and 99.95% ($\pm$2.80%)(standard deviations in brackets), respectively, although we adjusted k of kNN for the new training set size using cross-validation. The difference between the triplanar-CNN and the method from [71] with 120000 training voxels is statistically significant (Wilcoxon rank-sum test, $p = 8.6 \times 10^{-5}$).

Table 6.1: Comparison of methods applied for tibial cartilage segmentation. Acc. stands for accuracy, Sens. stands for sensitivity and Spec. stands for specificity

Method	Over 114 Scans	DSC	Acc.	Sens.	Spec.
Triplanar CNN	Mean	0.8249	99.93%	81.92%	99.97%
	Std. Dev.	4.26%	0.019%	7.62%	0.017%
State-of-the-art [71]	Mean	0.8135	99.92%	80.52%	99.96%
	Std. Dev.	4.87%	.02%	8.95%	0.02%



(a) Knee MRI slice **(b)** radiologist **(c)** triplanar CNN

Figure 6.2: MRI slice with segmentations by a radiologist and the proposed triplanar CNN. Our method is based on voxel classification, a 2D slice is taken from our 3D segmentation just for visualization.

The training took approximately 1.5 days on a 2.8 GHz processor and 16 GB RAM system, while classifying 2 million voxels took approximately 6 hours on the same system.

We also performed baseline experiments with 3D CNNs. With a $14 \times 14 \times 14$ 3D patch size and $3 \times 3 \times 3$ kernel size, the performance of a 3D CNN was worse (approximately 2% dip in DSC) than the triplanar CNN with patch size 28×28 and kernel size 5×5. Using these configurations, the training and testing times of the two approaches were comparable. Using a 3D CNN with 3D patch size $28 \times 28 \times 28$ (leading to 21952 elements and thus a 21952×21952 matrix to be inverted in the ZCA) and kernel size $5 \times 5 \times 5$ turned out to be computationally not feasible in our experimental setup as each iteration of back- and forward-propagation needed 10 to 15 times more time compared to our triplanar 2D CNN. We also experimented with a single 2D CNN, with patches extracted from just a single plane (the same which radiologists use for segmenting). However, the performance achieved by a single 2D CNN was inferior to that of the triplanar 2D CNN. Based on the above experiments we found that the triplanar CNN provides the best balance between computational costs and performance.

6.5 Discussion and Future Work

The method by Folkesson et al. [71] – in particular the features used, but also the classifier – has been carefully tailored towards the task. That we can perform better using a method that *learns* the image features using a deep learning architecture is the main result of our study. This is all the more remarkable because of the following differences in comparison to [71] and other CNN architectures:

- The features used by our method were extracted using only 2D kernels learned from three 2D CNNs. However, the method from [71] uses 3D kernels to extract features.
- In [71], each feature is calculated at 3 different scales. Our method relies on the features extracted using single scale CNNs.
- The number of training data points used by our method is just 120000, while in [71] more than 2100000 training data points are used.
- We used a logistic regression classifier (a linear classifier) compared to kNN (a non-linear classifier).
- We did not employ at any stage layer-wise pre-training of the CNN, although pre-training is often claimed to improve the results. We train our system of three CNNs in a single training process.

We plan to extend our study on the segmentation of tibial cartilage to the multiclass problem of segmenting bones and all cartilage compartments.

Chapter 7

Spatially Contextualized Convolutional Neural Network

7.1 Introduction

The use of contextual information is very important in image understanding [122]. One of the earliest works which used contextual information in image analysis is [123]. We have shown the application of a CNN based system for segmenting knee MRIs through voxel classification in Chapter 6. While training the CNN we take a few randomly selected patches, for which we take the label of the center pixel/voxel and patch data while constructing our training data-set. However, while doing that we throw away the contextual label information, i.e. the labels of the neighboring pixels/voxels.

Recalling the cost function of the 2D CNN (7.1), we can see that we are trying to learn the CNN parameters through maximizing the posterior probability associated with the true label in a log likelihood manner. However, this formulation does not take any contextual information into account and assumes that the labels of the neighboring pixels/voxels are independent of each other. It can lead us to a label field having rough and irregular object boundaries and spatial inconsistent label field.

This chapter will be a part of an under preparation article which will be an extension to our already published article in MICCAI 2013 [56].

$$Q_\lambda = \frac{1}{M}\sum_{m=1}^{M} -\ln(p(t^{(m)}|\boldsymbol{d}^{(0)^{(m)}};\boldsymbol{\Omega})) + \frac{\lambda}{2}\sum_{p=1}^{K}\sum_{q=1}^{S_O^{(L)}}\theta_{pq}^2 \tag{7.1}$$

Most of the approaches which use contextual information for image segmentation or use it for post processing to improve segmentation are MRF like approaches [124, 125, 110, 126, 127, 128]. When solving the labeling problem or applying post-processing using MRFs, we essentially solve an energy minimization problem for which the image energy is written as the sum of a data energy term and a smoothness energy term. To make it clearer, let $\boldsymbol{\omega}$ be the image labeling and ω_p be the label corresponding to an pixel/voxel p such that $\omega_p \in \boldsymbol{\omega}$. Also, $\boldsymbol{\mathcal{P}}$ be the set of all the pixels/voxels of the image and $\boldsymbol{\mathcal{N}}_p$ be the neighborhood of p. The image energy corresponding to labeling ($\boldsymbol{\omega}$) can be given as

$$U(\boldsymbol{\omega}) = \sum_{p\in\boldsymbol{\mathcal{P}}}\left(U_D(\omega_p) + \sum_{q\in\boldsymbol{\mathcal{N}}_p} U_S(\omega_q,\omega_p)\right), \tag{7.2}$$

where $U_D(\omega_p)$ is the data energy term for pixel p labeled as ω_p, while $U_S(\omega_q,\omega_p)$ is the smoothness energy term with pixel p labeled as ω_p while its neighboring pixel q labeled as ω_q. When used for post-processing to smooth the segmentation obtained from a pixel/voxel classifier , the data energy term ensures that the final labeling is coherent with the labeling assigned by the classifier while the smoothness energy term ensures that the final labeling is smooth, i.e. it penalize the cases where the two neighboring pixels/voxels have different labels.

If used as pixel classifiers themselves, the MRF models are used in a *Maximum a posteriori-Markov random field* framework where the features are already known [129]. MAP energy consists of a likelihood term and a smoothness prior term modeled with an MRF model.

However, in our case we neither have the features nor we apply MRF models for post-processing. Here we want to use contextual labels information for improving the learning of our CNN parameters (and thus the features) in order to obtain a smoother and spatially more consistent labeling (improved segmentation) in one step while performing segmentation. Taking inspiration from other MRF

based approaches, a possibility could have been to learn the network/features by modeling the posterior interactions using an MRF model. One such model is an Ising model. For the Ising model, the smoothness energy term $U_S(\omega_q, \omega_p)$ can be written as $b_{pq}\omega_p\omega_q$ where $b_{pq} < 0$ and neighborhood $\boldsymbol{\mathcal{N}}_p$ is a set of four nearest pixels in case of a 2D image. As $b_{pq} < 0$, the model penalizes the cases when $\omega_p \neq \omega_q$. Although it seems interesting to use such model for incorporating contextual information in CNN training, we couldn't do that as these models need the full lattice to be known, while for constructing the training data of the CNN, we randomly sample only few pixels/voxels from the training images.

The above reasons lead us to a novel solution which exploits the labels of the unused neighborhood voxels/pixels, which were not used while constructing our training dataset for CNN. For doing that, we add an extra term which measures the dissimilarity between the label posterior value at a voxel location and label configuration known around this voxel. Rest of the chapter is organized as follows. In the next section we introduce and discuss the cost function for training our spatially contextualized CNN(SCCNN). Then in section 7.3, we talk about optimizing our new cost function, following which we give experimental details and results. In the end we conclude the chapter in section 7.5

7.2 Learning Spatial Configuration

In this section we introduce and discuss our new cost function which takes the contextual neighborhood labels into consideration. To do that, we add to the CNN cost function a new term which will enforce the similarity between the posterior probability distribution for a pixel/voxel over all the classes and the class distribution around that pixel/voxel, where the class distributions around a sample are simply computed as relative frequencies of a class label in a simple neighborhood (7.3). In order to make it clearer, let N be the total number of voxels/pixels in the neighborhood in consideration (including the center voxel/pixel itself) and ${N_c}^{(m)}$ be the number of voxel/pixel belonging to class c in the neighborhood of mth training example. The fraction of neighborhood voxels/pixels belonging to class c is

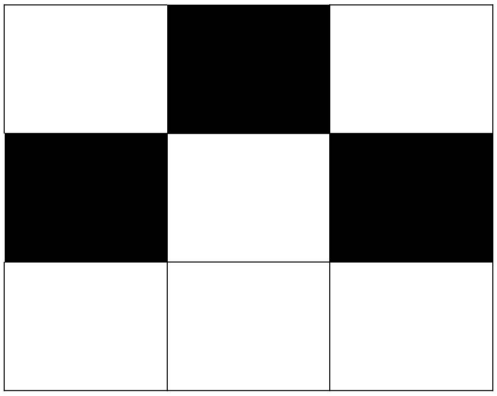


Figure 7.1: An example neighborhood configuration, black pixels are background pixels(class 2) and white pixels (class 1) are foreground pixels. $P_N(1)$ for center pixel in this case is 6/9 while $P_N(2)$ is 3/9

$$P_N^{(m)}(c) = {N_c}^{(m)}/N \tag{7.3}$$

Figure 7.1 shows an example neighborhood configuration for a 2D image and corresponding calculation of $P_N(c)$ for the center pixel.

Let ${\boldsymbol{P}_N}^{(m)}$ be the probability distribution formed by probabilities $P_N^{(m)}(c)$ over all the classes. Similarly, $\boldsymbol{p}^{(m)}$ be the probability distribution formed by probabilities $p(c|{\boldsymbol{d}^{(0)}}^{(m)}; \boldsymbol{\Omega})$ over all the classes. Now, an appropriate choice for our contextualization term will be a term which measures the dissimilarity between the two probability distribution functions. Thus our new cost function will be in the form shown below

$${Q_\lambda}^{(N)} = -\frac{1}{M}\sum_{m=1}^{M}\frac{\ln(p(t^{(m)}|{\boldsymbol{d}^{(0)}}^{(m)}; \boldsymbol{\Omega})) - \beta D(\boldsymbol{p}^{(m)}, {\boldsymbol{P}_N}^{(m)})}{1+\beta} + \frac{\lambda}{2}\sum_{p=1}^{K}\sum_{q=1}^{S_O^{(L)}}\theta_{pq}^2, \tag{7.4}$$

where $D(\boldsymbol{p}^{(m)}, {\boldsymbol{P}_N}^{(m)})$ is a term which is minimized if $\boldsymbol{p}^{(m)}$ is same as ${\boldsymbol{P}_N}^{(m)}$ and increases if $\boldsymbol{p}^{(m)}$ diverges away from ${\boldsymbol{P}_N}^{(m)}$.

Several such measures are found in the literature [130]e.g. Hellinger distance, Bhattacharya distance, Kullback–Leibler divergence etc. Let us have a brief look at them.

Hellinger distance for the our two probability distributions $\boldsymbol{p}^{(m)}$ and ${\boldsymbol{P}_N}^{(m)}$ is given as

$$D(\boldsymbol{p}^{(m)}, {\boldsymbol{P}_N}^{(m)}) = \frac{1}{\sqrt{2}} \sqrt{\sum_{c=1}^{K} \left(\sqrt{p(c|{\boldsymbol{d}^{(0)}}^{(m)}; \boldsymbol{\Omega})} - \sqrt{P_N^{(m)}(c)} \right)^2} \tag{7.5}$$

Another popular measure is **Bhattacharya distance** which is given as

$$D(\boldsymbol{p}^{(m)}, {\boldsymbol{P}_N}^{(m)}) = -\ln \left(\sum_{c=1}^{K} \sqrt{p(c|{\boldsymbol{d}^{(0)}}^{(m)}; \boldsymbol{\Omega}), P_N^{(m)}(c)} \right) \tag{7.6}$$

However, our contextualization term is inspired by the **Kullback–Leibler (KL) divergence** measure which is given as

$$D(\boldsymbol{p}^{(m)}, {\boldsymbol{P}_N}^{(m)}) = \sum_{c=1}^{K} P_N^{(m)}(c) \ln(\frac{P_N^{(m)}(c)}{p(c|{\boldsymbol{d}^{(0)}}^{(m)}; \boldsymbol{\Omega})}) \tag{7.7}$$

We chose KL divergence measure because it makes the new cost function easy to implement. Before going more into details, we write our new modified cost function for a 2D CNN by using equation (7.4) and equation (7.7)

$${Q_\lambda}^{(N)} = -\frac{1}{M} \sum_{m=1}^{M} \frac{\ln(p(t^{(m)}|{\boldsymbol{d}^{(0)}}^{(m)}; \boldsymbol{\Omega})) - \beta \sum_{c=1}^{K} P_N^{(m)}(c) \ln(\frac{P_N^{(m)}(c)}{p(c|{\boldsymbol{d}^{(0)}}^{(m)}; \boldsymbol{\Omega})})}{1+\beta} + \frac{\lambda}{2} \sum_{p=1}^{K} \sum_{q=1}^{S_O^{(L)}} \theta_{pq}^2 \tag{7.8}$$

The term $\sum_{c=1}^{K} P_N^{(m)}(c) \ln(\frac{P_N^{(m)}(c)}{p(c|{\boldsymbol{d}^{(0)}}^{(m)}; \boldsymbol{\Omega})})$, which is the KL divergence measure between $\boldsymbol{p}^{(m)}$ and ${\boldsymbol{P}_N}^{(m)}$ minimizes if $\boldsymbol{p}^{(m)}$ is same as ${\boldsymbol{P}_N}^{(m)}$ and increases monotonically if $\boldsymbol{p}^{(m)}$ diverges away from ${\boldsymbol{P}_N}^{(m)}$. β controls the strength of this term and $\beta = 0$ gives us the unmodified cost function. A close observation shows a strong connection between the log-likelihood term $\ln(p(t^{(m)}|{\boldsymbol{d}^{(0)}}^{(m)}; \boldsymbol{\Omega}))$ and our contextualization term $\beta \sum_{c=1}^{K} P_N^{(m)}(c) \ln(\frac{P_N^{(m)}(c)}{p(c|\boldsymbol{d}^{(0)}; \boldsymbol{\Omega})})$. For making it clearer, let $\boldsymbol{G}^{(m)}$ be the ground truth vector such that its c th element $G^{(m)}(c)$ is equal to one if the ground truth label $t^{(m)} = c$ and zero otherwise. Now, the log-likelihood term

$\ln(p(t^{(m)}|\boldsymbol{d}^{(0)^{(m)}};\boldsymbol{\Omega}))$ can be written in terms of $G^{(m)}(c)$ as $-\sum_{c=1}^{K} G^{(m)}(c)\ln(\frac{G^{(m)}(c)}{p(c|\boldsymbol{d}^{(0)^{(m)}};\boldsymbol{\Omega})})$. This similarity between the log-likelihood term and our contextualization term makes the new cost function easy to implement.

Similar to the 2D CNN case, the contextualized cost function for a triplanar CNN can be given as

$$Q_{T\lambda}{}^{(N)} = -\frac{1}{M}\sum_{m=1}^{M}\Bigg[\ln(p(t^{(m)}|\boldsymbol{d}_{xy}{}^{(0)^{(m)}};\boldsymbol{d}_{yz}{}^{(0)^{(m)}};\boldsymbol{d}_{zx}{}^{(0)^{(m)}};\boldsymbol{\Omega}_T)) - \beta\sum_{c=1}^{K} P_N^{(m)}(c)\ln(\frac{P_N^{(m)}(c)}{p(c|\boldsymbol{d}_{xy}^{(0)^{(m)}};\boldsymbol{d}_{yz}^{(0)^{(m)}};\boldsymbol{d}_{zx}^{(0)^{m}};\boldsymbol{\Omega}_T)})\Bigg]\Bigg/(1+\beta) + \frac{\lambda}{2}\sum_{p=1}^{K}\sum_{q=1}^{N_{OT}}\theta_{T_{pq}}^2 \quad (7.9)$$

7.3 Optimizing the New Extended Cost Function

Similar to the unmodified cost function, our new extended cost function is also optimized using LBFGS. The gradients used for that purpose are calculated as discussed in the following paragraph.

$\boldsymbol{\Omega}$ in equation (7.8), is the set of the parameters including all the CNN parameters as well as softmax parameters. Also the the output of the last layer $\boldsymbol{d}^{(L)^{(m)}}$ can be written in terms of CNN parameters and the input patch. Thus we can rewrite $p(t^{(m)}|\boldsymbol{d}^{(0)^{(m)}};\boldsymbol{\Omega})$ as $p(t^{(m)}|\boldsymbol{d}^{(L)^{(m)}};\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the softmax parameter matrix. Thus, the modified cost function in equation (7.8) can also be written as

$$Q_{\lambda}{}^{(N)} = -\frac{1}{M}\sum_{m=1}^{M}\frac{\ln(p(t^{(m)}|\boldsymbol{d}^{(L)^{(m)}};\boldsymbol{\theta})) - \beta\sum_{c=1}^{K} P_N^{(m)}(c)\ln(\frac{P_N^{(m)}(c)}{p(c|\boldsymbol{d}^{(L)^{(m)}};\boldsymbol{\theta})})}{1+\beta} + \frac{\lambda}{2}\sum_{p=1}^{K}\sum_{q=1}^{S_O^{(L)}}\theta_{pq}^2 \quad (7.10)$$

The gradient of the modified cost function with respect to softmax parameters can be easily calculated with the help of equation (5.11). The sensitivity map of

the final layer for CNN with modified cost function is calculated in a similar way as discussed in equation (5.13) and is given as

$$\boldsymbol{\delta}^{(L)^{(N)^{(m)}}} = -\boldsymbol{\theta}^T \boldsymbol{D}^{(N)^{(m)}} \tag{7.11}$$

However, $\boldsymbol{D}^{(N)^{(m)}}$ in this case is calculated using

$$\boldsymbol{D}^{(N)^{(m)}} = \begin{bmatrix} \frac{\mathbb{1}(t^{(m)},1) - p(t^{(m)}|\boldsymbol{\theta}_1;\boldsymbol{d}^{(L)^{(m)}}) + \beta(P_N^{(m)}(1) - p(t^{(m)}|\boldsymbol{\theta}_1;\boldsymbol{d}^{(L)^{(m)}}))}{1+\beta} \\ \frac{\mathbb{1}(t^{(m)},2) - p(t^{(m)}|\boldsymbol{\theta}_2;\boldsymbol{d}^{(L)^{(m)}}) + \beta(P_N^{(m)}(2) - p(t^{(m)}|\boldsymbol{\theta}_2;\boldsymbol{d}^{(L)^{(m)}}))}{1+\beta} \\ \vdots \\ \frac{\mathbb{1}(t^{(m)},c) - p(t^{(m)}|\boldsymbol{\theta}_c;\boldsymbol{d}^{(L)^{(m)}}) + \beta(P_N^{(m)}(c) - p(t^{(m)}|\boldsymbol{\theta}_c;\boldsymbol{d}^{(L)^{(m)}}))}{1+\beta} \\ \vdots \\ \frac{\mathbb{1}(t^{(m)},K) - p(t^{(m)}|\boldsymbol{\theta}_K;\boldsymbol{d}^{(L)^{(m)}}) + \beta(P_N^{(m)}(K) - p(t^{(m)}|\boldsymbol{\theta}_K;\boldsymbol{d}^{(L)^{(m)}}))}{1+\beta} \end{bmatrix} \tag{7.12}$$

where $\mathbb{1}(p,q)$ be a function returning one if $p = q$ and zero otherwise. The sensitivity maps and the gradients of rest of the CNN layers can be calculated in the same way as discussed in 5.5.

For triplanar CNN case, the modified cost function given in equation (7.9) can be re-written as

$$Q_{T\lambda}{}^{(N)} = -\frac{1}{M}\sum_{m=1}^{M} \frac{\ln(p(t^{(m)}|\boldsymbol{d}_T^{(L)^{(m)}};\boldsymbol{\theta}_T)) - \beta\sum_{c=1}^{K} P_N^{(m)}(c)\ln(\frac{P_N^{(m)}(c)}{p(c|\boldsymbol{d}_T^{(L)^{(m)}};\boldsymbol{\theta}_T)})}{1+\beta} + \frac{\lambda}{2}\sum_{p=1}^{K}\sum_{q=1}^{S_O^{(L)}}{\theta_T}_{pq}^2 \tag{7.13}$$

where $\boldsymbol{d}_T^{(L)^{(m)}}$ is a vector obtained by concatenating the outputs of three triplanar CNNs.

The gradient of the modified cost function with respect to softmax parameter can be calculated using (6.6).

The sensitivity maps of the final layer for CNN with modified cost function can be calculated using the following three equations

$$\boldsymbol{\delta}_{xy}{}^{(N)(L)^{(m)}} = -\boldsymbol{\theta}_{xy}{}^T \boldsymbol{D}^{(N)^{(m)}} \tag{7.14}$$

$$\boldsymbol{\delta}_{yz}{}^{(N)(L)(m)} = -\boldsymbol{\theta}_{yz}{}^{T}\boldsymbol{D}^{(N)(m)} \tag{7.15}$$

$$\boldsymbol{\delta}_{zx}{}^{(N)(L)(m)} = -\boldsymbol{\theta}_{zx}{}^{T}\boldsymbol{D}^{(N)(m)} \tag{7.16}$$

Vector $\boldsymbol{D}^{(N)(m)}$ in this case is given as

$$\boldsymbol{D}^{(N)(m)} = \begin{bmatrix} \frac{\mathbb{1}(t^{(m)},1)-p(t^{(m)}|\boldsymbol{\theta}_{T1};\boldsymbol{d}_T{}^{(L)(m)})+\beta(P_N^{(m)}(1)-p(t^{(m)}|\boldsymbol{\theta}_{T1};\boldsymbol{d}_T{}^{(L)(m)}))}{1+\beta} \\ \frac{\mathbb{1}(t^{(m)},1)-p(t^{(m)}|\boldsymbol{\theta}_{T2};\boldsymbol{d}_T{}^{(L)(m)})+\beta(P_N^{(m)}(2)-p(t^{(m)}|\boldsymbol{\theta}_{T2};\boldsymbol{d}_T{}^{(L)(m)}))}{1+\beta} \\ \vdots \\ \frac{\mathbb{1}(t^{(m)},1)-p(t^{(m)}|\boldsymbol{\theta}_{Tc};\boldsymbol{d}_T{}^{(L)(m)})+\beta(P_N^{(m)}(c)-p(t^{(m)}|\boldsymbol{\theta}_{Tc};\boldsymbol{d}_T{}^{(L)(m)}))}{1+\beta} \\ \vdots \\ \frac{\mathbb{1}(t^{(m)},1)-p(t^{(m)}|\boldsymbol{\theta}_{TK};\boldsymbol{d}_T{}^{(L)(m)})+\beta(P_N^{(m)}(K)-p(t^{(m)}|\boldsymbol{\theta}_{TK};\boldsymbol{d}_T{}^{(L)(m)}))}{1+\beta} \end{bmatrix} \tag{7.17}$$

As the three CNNs are only connected with each other in the last layer, we can calculate the the sensitivities maps and partial derivatives for three CNNs independently following the discussion in 5.5

7.4 Experiments

In order to evaluate the performances of the 2D SCCNN and the triplanar SCCNN, we used two different datasets. For the 2D SCCNN, we used the Weizmann Horses database [57], a database often use in Computer Vision research, especially for segmentation, and for the triplanar SCCNN, we used the knee MR data that has been extensively used too in the rest of this thesis.

7.4.1 Weizmann's Horses

The first data-set was Weizmann's horses [57] of 328 gray scale horses out of which 164 images were randomly selected to extract the training data. From the remaining 164 images, 30 images were randomly selected as test images.

Experimental Set-up We extracted patches from 366 randomly selected pixels from each of the training images. Thus, from 164 training images, we constructed

a training data-set of approximately 60000 patches. Patch-size and the kernel-size chosen were 28×28 and 5×5. We consider the 3×3 neighborhood for constructing the modified cost function. The sequence of layers of the CNN was $\mathrm{L_C} \rightarrow \mathrm{L_S} \rightarrow \mathrm{L_C} \rightarrow \mathrm{L_C} \rightarrow \mathrm{L_F}$. The sequence for number of output-maps was $28 \rightarrow 28 \rightarrow 56 \rightarrow 112 \rightarrow 1792$. We minimize ${Q_\lambda}^{(N)}$ (equation (7.8)) using LBFGS using mini-batches. For each mini-batch, we selected 2000 examples randomly from 60000 training examples and run 20 iterations of LBFGS. The motive behind experiments with Weizmann horses is to run some preliminary experiments on a 2D dataset and to check whether we can achieve better DSC using SCCNN or not. The layer sequence, layer size are taken directly from the configuration of a single CNN of our triplanar CNN. The experimental set-up used may not be the best for this data-set.

Results We evaluated the SCCNN in (7.8) on 30 test images and compared the results with cost function for standard CNN in (5.9). After classifying the voxels, we took the largest connected component to get the final segmentation. Figure 7.2 below shows the variation of the mean DSC as we change the value of β. The blue horizontal line shows the value of average DSC achieved by the triplanar CNN with unmodified cost function (i.e. $\beta = 0$). The values of β used to plot the average DSC in figure 7.2 are $2^{-14}, 2^{-12}, 2^{-10}, 2^{-8}, 2^{-6}, 2^{-4}, 2^{-2}$ The best performance was given by $\beta = 2^{-4}$, with average DSC being 0.7773, while the average DSC achieved by unmodified-cost function $\beta = 0$ was 0.7618. For ($\beta \leq 2^{-16}$), DSC values obtained were same as DSC values for unmodified cost function. Figure 7.3 shows an example test image and its segmentation obtained by the unmodified and modified cost functions.

7.4.2 Knee MRI Data

We also experimented with modified cost function for triplanar CNN (Equation: (7.9))on knee MRI scans to segment the tibial cartilage and compared the results with the unmodified cost for triplanar CNN (Equation: 6.2).

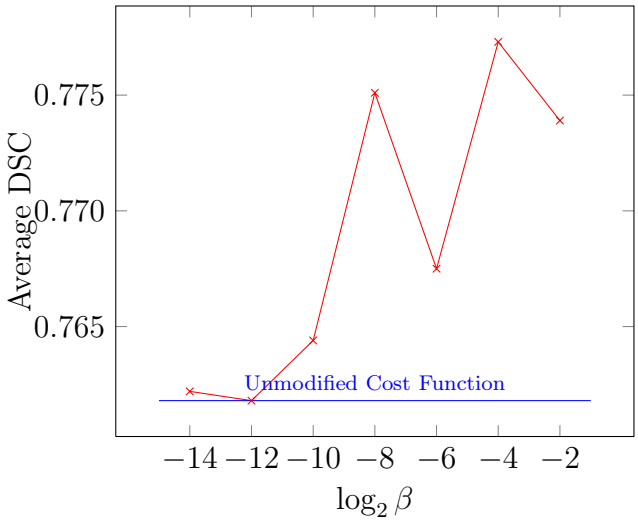
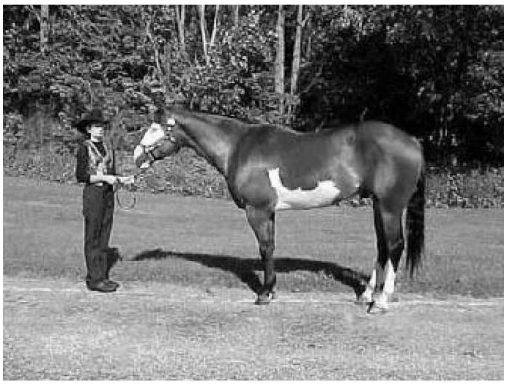


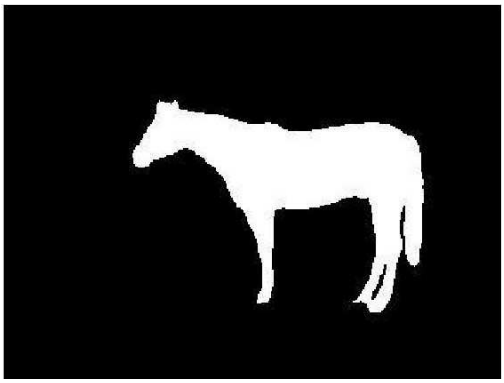
Figure 7.2: Average DSC values using our new extended cost function. Average of the DSC values over 30 horses' images are obtained for different values of β. The blue horizontal line depicts the Average DSC value obtained by unmodified cost function i.e. $\beta = 0$

Experimental Set-up Patch-size and the kernel-size which we used were 28×28 and 5×5, which are the same as mentioned in section 6.3. The sequence of layers of three CNNs was also the same, i.e. $\mathrm{L_C} \rightarrow \mathrm{L_S} \rightarrow \mathrm{L_C} \rightarrow \mathrm{L_C} \rightarrow \mathrm{L_F}$. Although for the triplanar CNN, we have three 2D patches as input, for constructing the contextualization term of the modified cost function, we took a neighborhood of $3 \times 3 \times 3$. The training data was constructed using the same voxels as mentioned in 6.3. The sequence of number of output-maps was also the same as mentioned in 6.3. We minimize ${Q_{T\lambda}}^{(N)}$ (equation (7.8)) using LBFGS using mini-batches. For each mini-batch, we selected 2000 examples randomly from 120000 training examples and run 20 iterations of LBFGS.

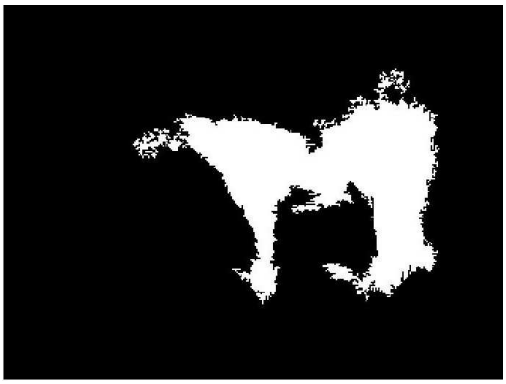
Results We performed test on 10 unseen scans. After classifying each voxel, the largest connected component was taken as final segmentation. Figure 7.4 shows the variation of the mean DSC as we change the value of β. The blue line show the value of average DSC achieved by the triplanar CNN with unmodified cost function (i.e. $\beta = 0$). The best performance was given by $\beta = 2^{-2}$, with average



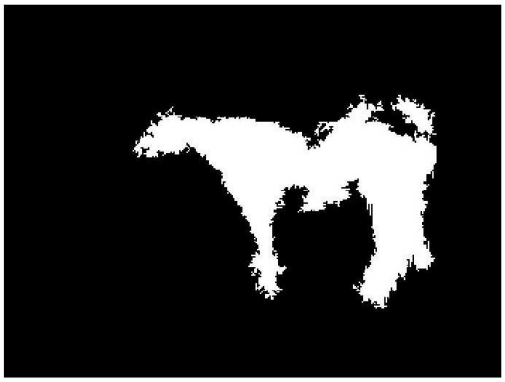
(a) gray scale horse image



(b) manual segmentation



(c) segmentation obtained using unmodified cost function, $\beta = 0$



(d) segmentation obtained using modified cost function, $\beta = 2^{-4}$

Figure 7.3: Test example comparing the segmentations obtained using unmodified and modified cost functions

DSC being 0.8100, while the average DSC achieved by unmodified-cost function $\beta = 0$ was 0.8030. Figure 7.5 shows a slice from a test scan and its segmentation obtained by the unmodified and modified cost functions. The results in Figure 7.4 are shown for $\beta = 2^{-8}, 2^{-6}, 2^{-4}, 2^{-2}$. The blue horizontal line depicts the Average DSC value obtained by unmodified cost function i.e. $\beta = 0$. For very small values of β ($\beta \leq 2^{-14}$), DSC values obtained were same as DSC values for unmodified cost function.

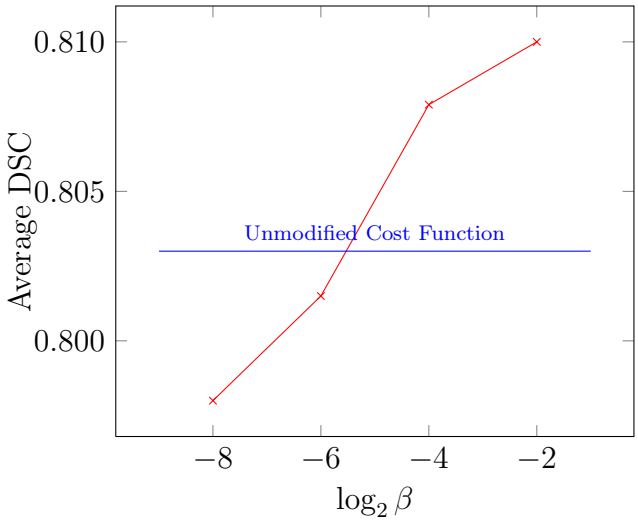
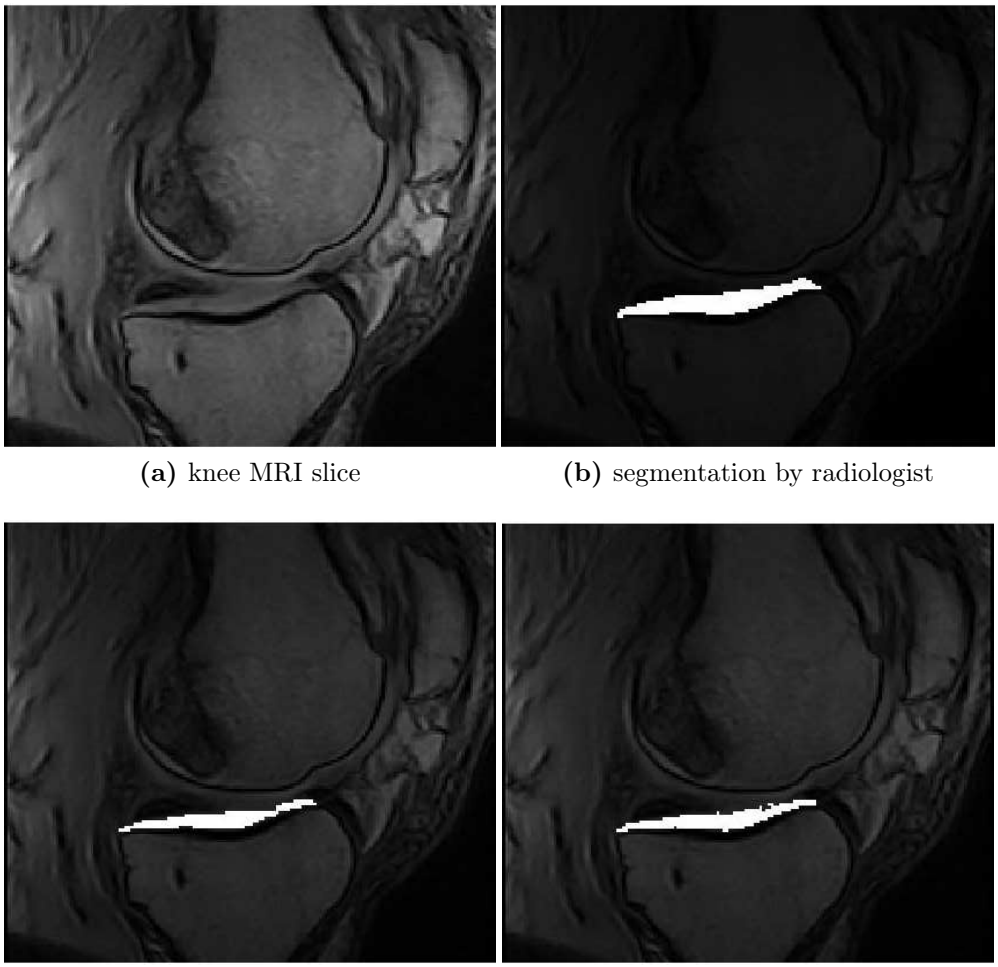


Figure 7.4: Average DSC values using our new extended cost function. Average of the DSC values over 10 MRI scans images are obtained for different values of β. The blue horizontal line depicts the Average DSC value obtained by unmodified cost function i.e. $\beta = 0$

7.5 Conclusion

The proposed Spatially Contextualized Convolutional Neural Network (SCCNN) incorporates the neighborhood label information to learn a network which improves the segmentation performance by making the output label field smoother and spatially more consistent. This is achieved by learning a better network while training and thus does not involve any post processing step while segmenting a new image. The cost function for our SCCNN has an extra term added to the cost function of standard CNN, which uses the KL divergence measure between the posterior probability distribution for a pixel/voxel over all the classes and the class distribution around that pixel/voxel, where the class distributions around a sample are simply computed as relative frequencies of a class label in a simple neighborhood. We evaluated our method on two data-sets of Weizmann horses and knee MRI scans. Our method improved the segmentation results in both the cases. In future we will perform more extensive experiments with more image datasets. We intend to compare various probability distribution measures including currently used KL



(a) knee MRI slice

(b) segmentation by radiologist

(c) segmentation obtained using unmodified cost function, $\beta = 0$

(d) segmentation obtained using modified cost function, $\beta = 2^{-2}$

Figure 7.5: Knee MRI slice from a test scan comparing the segmentations obtained using unmodified and modified cost functions

divergence measure. Though SCCNN was developed in order to incorporate spatial behavior in the learning phase, this should not prevent us to add an extra step of MRF-like post processing of the label fields. We plan to add this step in the future and compare the segmentation performances obtained with or without it for both the "standard" CNN and our proposed SCCNN.

Chapter 8

Discussion and Future Work

In this thesis we have presented voxel/pixel classification based methods for segmentation and showed there application mainly on segmenting articular cartilage from knee MRIs. We have addressed two of the main issues associated with voxel/pixel classification based segmentation problems. The first issue is related to the huge amount of training data needed, which restricts the usage of some very strong classifiers. The other issue we addressed is related to the features which should be used for the classification task.

In the first main part of the thesis, we proposed a cascaded classifier method. This method is a general tool for cases having huge amount of training data with large class imbalance. These situations are very commonly found in medical imaging and our method is particularly useful for such scenarios. Our cascaded classifier exploits the class imbalance and allows the usage of classifiers scaling badly with training data population, in large scale problems. We have shown its application on segmenting cartilage from knee MRIs, which was very successful. The training data had features from more than 2 million voxels with class populations having ratio of approximately 1:17. Such large-scale problem is difficult to handle by a classifier like non-linear SVM which scales at-least quadratically with the number of training data points.

Our two-stage method for cartilage segmentation, comprising first stage of kNN and second stage of SVM outperformed the state-of-the-art method based on one-

stage of kNN. The evaluation was done on 114 scans and the Wilcoxon rank-sum test for the DSC gave p-value less than 0.05. We tried to increase the speed of the SVM training by online SVM and also by loosening the stopping criterion in the quadratic programming solver. We found no advantage in using online SVMs over loosening the stopping condition in the quadratic program solver. We also tried to solve the problem in a single stage using online SVM, but found that it did not lead to better performance given our time budget.

Selecting good features is crucial in any classification problem including those in medical imaging. The next major part of this thesis deals with learning features from the data instead of having a predefined set of features. We proposed a deep learning method based on convolutional neural networks to classify voxels of a 3 dimensional image. Our method had 3 integrated CNNs, each corresponding to one of the three image planes, i.e. xy, yz and zx planes. We used it for segmenting tibial cartilage in knee MRI and the evaluation was performed on 114 scans. Although our method learnt 2D kernels from the three CNNs, it performed better than a state-of-the-art method which used predefined set of features extracted using 3D kernels. Moreover, the state-of-the-art method uses multi-scale features, whereas we relied on single scale features. The improvement over state-of-the-art were found to be statistically significant with p-value from Wilcoxon rank-sum test for DSC being less than 0.05.

While training the CNN, the cost function is simply based on the class-label of the voxel/pixel and does not take class-labels of the neighborhood voxels/pixels into consideration. We propose a spatially contextualized convolutional neural network (SCCNN) which incorporates the labels of the neighborhood voxels/pixels while learning the network. We evaluated 2D SCCNN on 30 scans from the Weizmann's horses dataset and also its triplanar variant (triplanar-SCCNN) on 10 knee MRI scans for segmenting tibial cartilage. The results were better than the standard 2D CNN and standard triplanar-CNN respectively. Though SCCNN was developed in order to incorporate spatial behaviour in the learning phase, this should not prevent us to add an extra step of MRF-like post processing of the label fields. We plan to add this step in the future and compare the segmentation

performances obtained with or without it for both the "standard" CNN and our proposed SCCNN.

Bibliography

[1] Igel, C.: Lecture Notes on Statistical Machine Learning. Department of Computer Science, University of Copenhagen (2013)

[2] Elnakib, A., Gimel'farb, G., Suri, J.S., El-Baz, A.: Medical image segmentation: a brief survey. In: Multi Modality State-of-the-Art Medical Image Segmentation and Registration Methodologies. Springer (2011) 1–39

[3] Suru, J.S., Wilson, D.L., Laximinarayan, S.: Handbook of biomedical image analysis. Volume 2. Springer (2005)

[4] Prastawa, M., Bullitt, E., Gerig, G.: Simulation of brain tumors in mr images for evaluation of segmentation efficacy. Medical image analysis **13**(2) (2009) 297–311

[5] El-Baz, A., Farag, A., Gimel'farb, G., Falk, R., El-Ghar, M.A., Eldiasty, T.: A framework for automatic segmentation of lung nodules from low dose chest CT scans. In: Pattern Recognition, 2006. ICPR 2006. 18th International Conference on. Volume 3., IEEE (2006) 611–614

[6] Greenspan, H., Ruf, A., Goldberger, J.: Constrained gaussian mixture model framework for automatic segmentation of MR brain images. Medical Imaging, IEEE Transactions on **25**(9) (2006) 1233–1245

[7] Ecabert, O., Peters, J., Schramm, H., Lorenz, C., von Berg, J., Walker, M.J., Vembar, M., Olszewski, M.E., Subramanyan, K., Lavi, G., et al.: Automatic model-based segmentation of the heart in CT images. Medical Imaging, IEEE Transactions on **27**(9) (2008) 1189–1201

[8] El-Baz, A., Farag, A.A., Yuksel, S.E., El-Ghar, M.E., Eldiasty, T.A., Ghoneim, M.A.: Application of deformable models for the detection of acute renal rejection. In: Deformable Models. Springer (2007) 293–333

[9] Pham, D.L., Xu, C., Prince, J.L.: Current methods in medical image segmentation 1. Annual review of biomedical engineering **2**(1) (2000) 315–337

[10] Polakowski, W.E., Cournoyer, D.A., Rogers, S.K., DeSimio, M.P., Ruck, D.W., Hoffmeister, J.W., Raines, R.A.: Computer-aided breast cancer detection and diagnosis of masses using difference of gaussians and derivative-based feature saliency. Medical Imaging, IEEE Transactions on **16**(6) (1997) 811–819

[11] Cheng, H.D., Lui, Y.M., Freimanis, R.I.: A novel approach to microcalcification detection using fuzzy logic technique. Medical Imaging, IEEE Transactions on **17**(3) (1998) 442–450

[12] Li, S.Z.: Markov random field modeling in computer vision. Springer-Verlag New York, Inc. (1995)

[13] Lee, C., Huh, S., Ketter, T.A., Unser, M.: Unsupervised connectivity-based thresholding segmentation of midsagittal brain mr images. Computers in biology and medicine **28**(3) (1998) 309–338

[14] Gibbs, P., Buckley, D., Blackband, S., Horsman, A.: Tumour volume determination from mr images by morphological segmentation. Physics in medicine and biology **41**(11) (1996) 2437

[15] Pohlman, S., Powell, K.A., Obuchowski, N.A., Chilcote, W.A., Grundfest-Broniatowski, S.: Quantitative classification of breast tumors in digitized mammograms. Medical Physics **23** (1996) 1337

[16] Mangin, J.F., Frouin, V., Bloch, I., Régis, J., López-Krahe, J.: From 3d magnetic resonance images to structural representations of the cortex topography using topology preserving deformations. Journal of Mathematical Imaging and Vision **5**(4) (1995) 297–318

[17] Udupa, J.K., Samarasekera, S.: Fuzzy connectedness and object definition: theory, algorithms, and applications in image segmentation. Graphical models and image processing **58**(3) (1996) 246–261

[18] Wells III, W.M., Grimson, W.E.L., Kikinis, R., Jolesz, F.A.: Adaptive segmentation of mri data. Medical Imaging, IEEE Transactions on **15**(4) (1996) 429–442

[19] Vannier, M.W., Butterfield, R.L., Jordan, D., Murphy, W.A., Levitt, R.G., Gado, M.: Multispectral analysis of magnetic resonance images. Radiology **154**(1) (1985) 221–224

[20] Kapur, T., Eric, W., Grimson, L., Kikinis, R., Wells, W.M.: Enhanced spatial priors for segmentation of magnetic resonance imagery. In: Medical Image Computing and Computer-Assisted Interventation—MICCAI'98. Springer (1998) 457–468

[21] Rajapakse, J.C., Giedd, J.N., Rapoport, J.L.: Statistical approach to segmentation of single-channel cerebral mr images. Medical Imaging, IEEE Transactions on **16**(2) (1997) 176–186

[22] Pham, D.L., Prince, J.L.: An adaptive fuzzy< i> c</i>-means algorithm for image segmentation in the presence of intensity inhomogeneities. Pattern Recognition Letters **20**(1) (1999) 57–68

[23] Held, K., Kops, E.R., Krause, B.J., Wells III, W.M., Kikinis, R., Muller-Gartner, H.W.: Markov random field segmentation of brain mr images. Medical Imaging, IEEE Transactions on **16**(6) (1997) 878–886

[24] Chen, C., Lee, G.: On digital mammogram segmentation and microcalcification detection using multiresolution wavelet analysis. Graphical Models and Image Processing **59**(5) (1997) 349–364

[25] Bezdek, J.C., Hall, L., Clarke, L.: Review of mr image segmentation techniques using pattern recognition. Medical physics **20** (1993) 1033

[26] Hall, L.O., Bensaid, A.M., Clarke, L.P., Velthuizen, R.P., Silbiger, M.S., Bezdek, J.C.: A comparison of neural network and fuzzy clustering techniques in segmenting magnetic resonance images of the brain. Neural Networks, IEEE Transactions on **3**(5) (1992) 672–682

[27] Gelenbe, E., Feng, Y., Krishnan, K.R.R.: Neural network methods for volumetric magnetic resonance imaging of the human brain. Proceedings of the IEEE **84**(10) (1996) 1488–1496

[28] Reddick, W.E., Glass, J.O., Cook, E.N., Elkin, T.D., Deaton, R.J.: Automated segmentation and classification of multispectral magnetic resonance images of brain using artificial neural networks. Medical Imaging, IEEE Transactions on **16**(6) (1997) 911–918

[29] Vilariño, D.L., Brea, V.M., Cabello, D., Pardo, J.: Discrete-time cnn for image segmentation by active contours. Pattern Recognition Letters **19**(8) (1998) 721–734

[30] Davatzikos, C., Bryan, N.: Using a deformable surface model to obtain a shape representation of the cortex. Medical Imaging, IEEE Transactions on **15**(6) (1996) 785–795

[31] McInerney, T., Terzopoulos, D.: Medical image segmentation using topologically adaptable surfaces. In: CVRMed-MRCAS'97, Springer (1997) 23–32

[32] Xu, C., Pham, D.L., Rettmann, M.E., Yu, D.N., Prince, J.L.: Reconstruction of the human cerebral cortex from magnetic resonance images. Medical Imaging, IEEE Transactions on **18**(6) (1999) 467–480

[33] Bardinet, E., Cohen, L.D., Ayache, N.: A parametric deformable model to fit unstructured 3d data. Computer vision and image understanding **71**(1) (1998) 39–54

[34] Neumann, A., Lorenz, C.: Statistical shape model based segmentation of medical images. Computerized Medical Imaging and Graphics **22**(2) (1998) 133–143

[35] Mikic, I., Krucinski, S., Thomas, J.D.: Segmentation and tracking in echocardiographic sequences: Active contours guided by optical flow estimates. Medical Imaging, IEEE Transactions on **17**(2) (1998) 274–284

[36] Collins, D.L., Holmes, C.J., Peters, T.M., Evans, A.C.: Automatic 3-d model-based neuroanatomical segmentation. Human Brain Mapping **3**(3) (1995) 190–208

[37] Davatzikos, C.: Spatial normalization of 3d brain images using deformable models. Journal of computer assisted tomography **20**(4) (1996) 656–665

[38] Christensen, G.E., Joshi, S.C., Miller, M.I.: Volumetric transformation of brain anatomy. Medical Imaging, IEEE Transactions on **16**(6) (1997) 864–877

[39] Aboutanos, G.B., Dawant, B.M.: Automatic brain segmentation and validation: image-based versus atlas-based deformable models. In: Medical Imaging, SPIE Proc. Volume 3034. (1997) 299–310

[40] Pathak, S.D., Grimm, P.D., Chalana, V., Kim, Y.: Pubic arch detection in transrectal ultrasound guided prostate cancer therapy. Medical Imaging, IEEE Transactions on **17**(5) (1998) 762–771

[41] Bae, K.T., Giger, M.L., Chen, C.T., Kahn Jr, C.E.: Automatic segmentation of liver structure in ct images. Medical physics **20** (1993) 71

[42] Vincent, L., Soille, P.: Watersheds in digital spaces: an efficient algorithm based on immersion simulations. IEEE transactions on pattern analysis and machine intelligence **13**(6) (1991) 583–598

[43] Sijbers, J., Scheunders, P., Verhoye, M., Van der Linden, A., van Dyck, D., Raman, E.: Watershed-based segmentation of 3d mr data for volume quantization. Magnetic Resonance Imaging **15**(6) (1997) 679–688

[44] Vos, T., Flaxman, A.D., Naghavi, M., Lozano, R., Michaud, C., Ezzati, M., Shibuya, K., Salomon, J.A., Abdalla, S., Aboyans, V., et al.: Years lived

with disability (ylds) for 1160 sequelae of 289 diseases and injuries 1990–2010: a systematic analysis for the global burden of disease study 2010. The Lancet **380**(9859) (2013) 2163–2196

[45] Levit, K., Wier, L., Stranges, E., Ryan, K., Elixhauser, A.: Hcup facts and figures: statistics on hospital-based care in the united states, 2007. Rockville, MD: Agency for Healthcare Research and Quality (2009)

[46] Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning **20**(3) (1995) 273–297

[47] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (1998) 2278–2324

[48] Xiang, S., Nie, F., Zhang, C.: Learning a mahalanobis distance metric for data clustering and classification. Pattern Recognition **41**(12) (2008) 3600–3612

[49] J., G., S., R., G., H., R., S.: Neighborhood components analysis. In: Advances in Neural Information Processing Systems. (2004) 513–520

[50] Blitzer, J., Weinberger, K.Q., Saul, L.K.: Distance metric learning for large margin nearest neighbor classification. In: Advances in neural information processing systems. (2005) 1473–1480

[51] Globerson, A., Roweis, S.T.: Metric learning by collapsing classes. In: Advances in neural information processing systems. (2005) 451–458

[52] Bentley, J.L.: Multidimensional divide-and-conquer. Communications of the ACM **23**(4) (1980) 214–229

[53] Friedman, J.H., Bentley, J.L., Finkel, R.A.: An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software (TOMS) **3**(3) (1977) 209–226

[54] Prasoon, A., Igel, C., Loog, M., Lauze, F., Dam, E., Nielsen, M.: Cascaded classifier for large-scale data applied to automatic segmentation of articular cartilage. In: Proceeding of SPIE Medical Imaging: Image Processing. Volume 8314. (2012)

[55] Prasoon, A., Igel, C., Loog, M., Lauze, F., Dam, E., Nielsen, M.: Femoral cartilage segmentation in knee MRI scans using two stage voxel classification. In: Engineering in Medicine and Biology Society (EMBC), 35th Annual International Conference of the IEEE. (2013) 5469–5472

[56] Prasoon, A., Petersen, K., Igel, C., Lauze, F., Dam, E., Nielsen, M.: Deep feature learning for knee cartilage segmentation using a triplanar convolutional Neural Network. In: Medical Image Computing and Computer-Assisted Intervention. Volume 8150., Lecture Notes in Computer Science, Springer Science+Business Media 2013 (2013) 246–253

[57] Borenstein, E., Ullman, S.: Class-specific, top-down segmentation. In: Computer Vision—European Conference on Computer Vision 2002. Springer (2002) 109–122

[58] Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press (2002)

[59] Steinwart, I.: Sparseness of support vector machines. Journal of Machine Learning Research **4** (2003) 1071–1105

[60] Bottou, L., Lin, C.J.: Support vector machine solvers. In Bottou, L., Chapelle, O., DeCoste, D., Weston, J., eds.: Large Scale Kernel Machines. MIT Press, Cambridge, MA. (2007) 301–320

[61] Osuna, E., Freund, R., Girosi, F.: Improved training algorithm for support vector machines. In Principe, J., Giles, L., Morgan, N., Wilson, E., eds.: Neural Networks for Signal Processing VII, IEEE Press (1997) 276–285

[62] Platt, J.: Fast training of support vector machines using sequential minimal optimization. In Schölkopf, B., Burges, C.J.C., Smola, A.J., eds.: Advances in Kernel Methods - Support Vector Learning. MIT Press (1999) 185–208

[63] Joachims, T.: Making large-scale SVM learning practical. In Sch B., ed.: Advances in Kernel Methods – Support Vector Learning. MIT Press (1999) 169–184

[64] Lin, C.J.: On the convergence of the decomposition method for support vector machines. IEEE Transactions on Neural Networks **12** (2001) 1288–1298

[65] Keerthi, S.S., Gilbert, E.G.: Convergence of a generalized SMO algorithm for SVM classifier design. Machine Learning **46** (2002) 351–360

[66] Hush, D., Scovel, C.: Polynomial-time decomposition algorithms for support vector machines. Machine Learning **51** (2003) 51–71

[67] Fan, R.E., Chen, P.H., Lin, C.J.: Working set selection using the second order information for training support vector machines. Journal of Machine Learning Research **6** (2005) 1889–1918

[68] Glasmachers, T., Igel, C.: Maximum-gain working set selection for support vector machines. Journal of Machine Learning Research **7** (2006) 1437–1466

[69] List, N.: Convergence Rates for SVM-Decomposition-Algorithms. Doctoral thesis, Department of Mathematics, Ruhr-Universität Bochum (2008)

[70] List, N., Simon, H.U.: SVM-optimization and steepest-descent line search. In: Proceedings of the 21st Annual Conference on Learning Theory (COLT 2009). (2009) Submitted.

[71] Folkesson, J., Dam, E., Olsen, O., Pettersen, P., Christiansen, C.: Segmenting articular cartilage automatically using a voxel classification approach. IEEE Transactions on Medical Imaging **26**(1) (2007) 106–115

[72] Lin, H.T., Lin, C.J.: A study on sigmoid kernels for svm and the training of non-psd kernels by smo-type methods. submitted to Neural Computation (2003) 1–32

[73] Jackson, D., Simon, T., Aberman, H.: Symptomatic articular cartilage degeneration: The impact in the new millenium. Clinical Orthopaedics and Related Research **133** (2001) 14–25

[74] Helmick, C.G., Felson, D.T., Lawrence, R.C., Gabriel, S., Hirsch, R., Kwoh, C.K., Liang, M.H., Kremers, H.M., Mayes, M., Merkel, P.A., Pillemer, S.R., Reveille, J.D., Stone, J.H.: Estimates of the prevalence of arthritis and other rheumatic conditions in the united states. Arthritis and Rheumatism **58** (2008) 26–35

[75] Graichen, H., Eisenhart-Rothe, R., Vogl, T., Englmeier, K.H., Eckstein, F.: Quantitative assessment of cartilage status in osteoarthritis by quantitative magnetic resonance imaging. Arthritis Rheumatism **50**(3) (Mar. 2001) 811–816

[76] Stammberger, T., Eckstein, F., Michaelis, M., Englmeier, K.H., Reiser, M.: Interobserver reproducibility of quantitative cartilage measurements: Comparison of b-spline snakes and manual segmentation. Magnetic Resonance Imaging **17**(7) (1999) 1033–1042

[77] Lynch, J.A., Zaim, S., Zhao, J., Stork, A., Peterfy, C.G., Genant, H.K.: Cartilage segmentation of 3-D MRI scans of the osteoarthritic knee combining user knowledge and active contours. In: Proceeding of SPIE Medical Imaging 2000: Image Processing. Volume 3979. (2000) 925–935

[78] Solloway, S., Hutchinson, C., Vaterton, J., Taylor, C.: The use of active shape models for making thickness measurements of articular cartilage from MR images. Magnetic Resonance in Medicine **37** (1997) 943–952

[79] Pakin, S.K., Tamez-Pena, J.G., Totterman, S., Parker, K.J.: Segmentation, surface extraction and thickness computation of articular cartilage. In:

Proceeding of SPIE Medical Imaging 2002: Image Processing. Volume 4684. (2002) 155–166

[80] Grau, V., Mewes, A., Alcaiz, M., Kikinis, R., Warfield, S.: Improved watershed transform for medical image segmentation using prior information. IEEE Transactions on Medical Imaging **23**(4) (2004) 447–458

[81] Warfield, S.K., Winalski, C., Jolesz, F.A., Kikinis, R.: Automatic segmentation of MRI of the knee, Sydney, Australia, ISMRM Sixth Scientific Meeting and Exhibition (1998) 563

[82] Warfield, S.K., Kaus, M., Jolesz, F.A., Kikinis, R.: Adaptive, template moderated, spatially varying statistical classification. Medical Image Analysis **23**(4) (2000) 43–55

[83] Bae, K., Shim, H., Tao, C., Chang, S., Wang, J., Boudreau, R., Kwoh, C.: Intra and inter-observer reproducibility of volume measurement of knee cartilage segmented from the OAI MR image set using a novel semi-automated segmentation method. Osteoarthritis and Cartilage **17**(12) (2009) 1589 – 1597

[84] Chang, K.Y., Chen, S.J., Chen, L.S., Wu, C.J.: Articular cartilage segmentation based on radial transformation. In: 9th International Conference on Hybrid Intelligent Systems (HIS 2009), August 12-14, 2009, Shenyang, China, IEEE Computer Society (2009) 239–242

[85] Yin, Y., Zhang, X., Anderson, D.D., Brown, T.D., Hofwegen, C.V., Sonka, M.: Simultaneous segmentation of the bone and cartilage surfaces of a knee joint in 3D. In: Proceeding of SPIE Medical Imaging 2009: Physics of Medical Imaging. Volume 7258. (2009) 72591O–72591O–9

[86] Seim, H., Kainmueller, D., Lamecker, H., Bindernagel, M., Malinowski, J., Zachow, S.: Model-based auto-segmentation of knee bones and cartilage in MRI data. In: Proceeding of Medical Image Analysis for the Clinic: A Grand Challenge. Bejing, China. (2010) 215 – 223

[87] Vincent, G., Wolstenholme, C., Scott, I., Bowes, M.: Fully automatic segmentation of the knee joint using active appearance models. In: Proceeding of Medical Image Analysis for the Clinic: A Grand Challenge. (2010) 224 –230

[88] Hinrichs, E., Appleton, B., Lovell, B., Galloway, G.: Autonomous direct 3D segmentation of articular knee cartilage. In: Australian and New Zealand Intelligent Information Systems. Volume 1., Queensland University of Technology (2011) 417 – 420

[89] Fripp, J., Crozier, S., Warfield, S., Ourselin, S.: Automatic segmentation and quantitative analysis of the articular cartilages from magnetic resonance images of the knee. IEEE Transactions on Medical Imaging **29**(1) (January 2010) 55–64

[90] Dodin, P., Pelletier, J., Martel-Pelletier, J., Abram, F.: Automatic human knee cartilage segmentation from 3-D magnetic resonance images. IEEE Transactions on Biomedical Engineering **57**(11) (2010) 2699 – 2711

[91] Lee, S., Park, S.H., Shim, H., Yun, I.D., Lee, S.U.: Optimization of local shape and appearance probabilities for segmentation of knee cartilage in 3-d mr images. Computer Vision and Image Understanding **115**(12) (2011) 1710–1720

[92] Shan, L., Charles, C., Niethammer, M.: Automatic multi-atlas-based cartilage segmentation from knee mr images. In: Biomedical Imaging (ISBI), 2012 9th IEEE International Symposium on, IEEE (2012) 1028–1031

[93] Wang, Q., Wu, D., Lu, L., Liu, M., Boyer, K., Zhou, S.: Semantic context forests for learning-based knee cartilage segmentation in 3d mr images. In: Medical Computer Vision. Large Data in Medical Imaging. Lecture Notes in Computer Science. Springer International Publishing (2014) 105–115

[94] Fripp, J., Crozier, S., Warfield, S.K., Ourselin, S.: Automatic segmentation and quantitative analysis of the articular cartilages from magnetic resonance

images of the knee. Medical Imaging, IEEE Transactions on **29**(1) (2010) 55–64

[95] Yin Yin; Xiangmin Zhang; Williams, R.; Xiaodong Wu; Anderson, D.S.M.: LOGISMOS - layered optimal graph image segmentation of multiple objects and surfaces: Cartilage segmentation in the knee joint. IEEE Transactions on Medical Imaging **29**(12) (2010) 2023 – 2037

[96] Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), IEEE Computer Society (2001) 511–518

[97] Viola, P., Jones, M.: Robust real-time object detection. International Journal of Computer Vision **57**(2) (2002) 137–154

[98] Batra, D., Singhal, G., Chaudhury, S.: Gabor filter based fingerprint classification using support vector machines. In: Proceedings of the First IEEE India Annual Conference (INDICON 2004), IEEE (2004) 256–261

[99] Koenderink, J.J.: The structure of images. Biological Cybernetics **50** (1984) 363–370

[100] Jaakkola, T., Diekhaus, M., Haussler, D.: Using the Fisher kernel method to detect remote protein homologies. In: Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology. (1999) 149–158

[101] Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2**(3) (2011) 27:1–27:27

[102] Zou, K.H., Warfield, S.K., Bharatha, A., Tempany, C.M., Kaus, M.R., Haker, S.J., Wells III., W.M., Jolesz, F.A., Kikinis, R.: Statistical validation of image segmentation quality based on a spatial overlap index. Academic Radiology **11** (2004) 178–189

[103] Arya, S., Mount, D., Netanyahu, N., Silverman, R., Wu, A.: An optimal algorithm for approximate nearest neighbor searching fixed dimensions. Journal of the ACM **45**(6) (1998) 891– 923

[104] Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research **9** (2008) 1871–1874

[105] Bordes, A., Ertekin, S., Weston, J., Bottou, L.: Fast kernel classifiers with online and active learning. The Journal of Machine Learning Research **6** (2005) 1579–1619

[106] Glasmachers, T., Igel, C.: Second order SMO improves SVM online and active learning. Neural Computation **20**(2) (2008) 374–382

[107] Lin, C.J.: A formal analysis of stopping criteria of decomposition methods for support vector machines. IEEE Transactions on Neural Networks **13**(5) (2002) 1045–1052

[108] Bordes, A., Ertekin, S., Weston, J., Bottou, L.: Fast kernel classifiers with online and active learning. Journal of Machine Learning Research **6** (2005) 1579–1619

[109] Rosenblatt, F.: Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. 1962. Washington DC: Spartan

[110] Ning, F., Delhomme, D., LeCun, Y., Piano, F., Bottou, L., Barbano, P.E.: Toward automatic phenotyping of developing embryos from videos. IEEE Transactions on Image Processing **14**(9) (2005) 1360–1371

[111] Sermanet, P., LeCun, Y.: Traffic sign recognition with multi-scale convolutional networks. In: Proceedings of International Joint Conference on Neural Networks (IJCNN'11). (2011) 2809–2813

[112] Schulz, H., Behnke, S.: Object-class segmentation using deep convolutional neural networks. In: DAGM Workshop on New Challenges in Neural Computation. (2011)

[113] Ciresan, D.C., Meier, U., Masci, J., Gambardella, L.M., Schmidhuber, J.: Flexible, high performance convolutional neural networks for image classification. In: IJCAI. (2011) 1237–1242

[114] Ciresan, D., Giusti, A., Schmidhuber, J., et al.: Deep neural networks segment neuronal membranes in electron microscopy images. In: Advances in Neural Information Processing Systems 25. (2012) 2852–2860

[115] Ji, S., Xu, W., Yang, M., Yu, K.: 3D convolutional neural networks for Human action recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(1) (2013) 221–231

[116] Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: International Conference on Document Analysis and Recognition. (2003) 958–962

[117] Le, Q.V., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., Ng, A.Y.: On optimization methods for deep learning. In: International Conference on Machine Learning. (2011) 265–272

[118] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. MIT Press, Cambridge, MA, USA (1988)

[119] Schulz, H., Behnke, S.: Learning object-class segmentation with convolutional neural networks. In: European Symposium on Artificial Neural Networks, Brussels, Belgium (2012)

[120] Turaga, S.C., Murray, J.F., Jain, V., Roth, F., Helmstaedter, M., Briggman, K.L., Denk, W., Seung, H.S.: Convolutional networks can learn to generate affinity graphs for image segmentation. Neural Computation **22**(2) (2010) 511–538

[121] Wang, T., Wu, D.J., Coates, A., Ng, A.Y.: End-to-end text recognition with convolutional neural networks. In: International Conference on Pattern Recognition. (2012) 3304–3308

[122] Pavlidis, T.: A critical survey of image analysis methods. In: International Conference on Pattern Recognition. (1986) 502–511

[123] Chow, C.K.: A recognition method using neighbor dependence. (1962) 683–690

[124] Boykov, Y., Kolmogorov, V.: Computing geodesics and minimal surfaces via graph cuts. In: Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on, IEEE (2003) 26–33

[125] Rother, C., Kolmogorov, V., Blake, A.: Grabcut: Interactive foreground extraction using iterated graph cuts. In: ACM Transactions on Graphics (TOG). Volume 23., ACM (2004) 309–314

[126] Boykov, Y., Funka-Lea, G.: Graph cuts and efficient nd image segmentation. International Journal of Computer Vision **70**(2) (2006) 109–131

[127] Singaraju, D., Grady, L., Vidal, R.: P-brush: Continuous valued mrfs with normed pairwise distributions for image segmentation. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE (2009) 1303–1310

[128] Malan, D.F., Botha, C.P., Valstar, E.R.: Voxel classification and graph cuts for automated segmentation of pathological periprosthetic hip anatomy. International Journal of Computer Assisted Radiology and Surgery **8**(1) (2013) 63–74

[129] Geman, S., Geman, D.: Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. Pattern Analysis and Machine Intelligence, IEEE Transactions on (6) (1984) 721–741

[130] Dodge, Y.: The Oxford dictionary of statistical terms. Oxford University Press (2006)