

University of Kaiserslautern
Department of Computer Science

Image Analysis with Long Short-Term Memory Recurrent Neural Networks

by

Wonmin Byeon

Thesis approved by the
Department of Computer Science
University of Kaiserslautern (TU Kaiserslautern)
for the award of the doctoral degree
Doctor of Natural Sciences (Dr. rer. nat.)

Date of submission: 02 December, 2015

Date of the viva: 07 March, 2016

Dean: Prof. Dr. Klaus Schneider

PhD committee

Chairperson: Prof. Dr. Katharina Zweig

Reviewers: Prof. Dr. Andreas Dengel

apl. Prof. Dr. Marcus Eichenberger-Liwicki

D386

Abstract

Computer Vision (CV) problems, such as image classification and segmentation, have traditionally been solved by manual construction of feature hierarchies or incorporation of other prior knowledge. However, noisy images, varying viewpoints and lighting conditions of images, and clutters in real-world images make the problem challenging. Such tasks cannot be efficiently solved without learning from data. Therefore, many Deep Learning (DL) approaches have recently been successful for various CV tasks, for instance, image classification, object recognition and detection, action recognition, video classification, and scene labeling. The main focus of this thesis is to investigate a purely learning-based approach, particularly, Multi-Dimensional LSTM (MD-LSTM) recurrent neural networks to tackle the challenging CV tasks, classification and segmentation on 2D and 3D image data. Due to the structural nature of MD-LSTM, the network learns directly from raw pixel values and takes the complex spatial dependencies of each pixel into account. This thesis provides several key contributions in the field of CV and DL.

Several MD-LSTM network architectural options are suggested based on the type of input and output, as well as the requiring tasks. Including the main layers, which are an input layer, a hidden layer, and an output layer, several additional layers can be added such as a collapse layer and a fully connected layer. First, a single *Two Dimensional LSTM (2D-LSTM)* is directly applied on *texture images for segmentation* and show improvement over other texture segmentation methods. Besides, a 2D-LSTM layer with a collapse layer is applied for *image classification on texture and scene images* and have provided an accurate classification results. In addition, a deeper model with a fully connected layer is introduced to deal with more complex images for *scene labeling* and outperforms the other state-of-the-art methods including the deep Convolutional Neural Networks (CNN). Here, several input and output representation techniques are introduced to achieve the robust classification. Randomly sampled windows as input are transformed in scaling and rotation, which are integrated to get the final classification. To achieve multi-class image classification

on scene images, several pruning techniques are introduced. This framework provides a good results in *automatic web-image tagging*. The next contribution is an investigation of 3D data with MD-LSTM. The traditional cuboid order of computations in Multi-Dimensional LSTM (MD-LSTM) is re-arranged in pyramidal fashion. The resulting *Pyramidal Multi-Dimensional LSTM (PyraMiD-LSTM)* is easy to parallelize, especially for 3D data such as stacks of brain slice images. PyraMiD-LSTM was tested on *3D biomedical volumetric images* and achieved best known pixel-wise brain image segmentation results and competitive results on Electron Microscopy (EM) data for membrane segmentation.

To validate the framework, several challenging databases for classification and segmentation are proposed to overcome the limitations of current databases. First, scene images are randomly collected from the web and used for scene understanding, i.e., *the web-scene image dataset* for multi-class image classification. To achieve multi-class image classification, the training and testing images are generated in a different setting. For training, images belong to a single pre-defined category which are trained as a regular single-class image classification. However, for testing, images containing multi-classes are randomly collected by web-image search engine by querying the categories. All scene images include noise, background clutter, unrelated contents, and also diverse in quality and resolution. This setting can make the database possible to evaluate for real-world applications. Secondly, an *automated blob-mosaics texture dataset generator* is introduced for segmentation. Random 2D Gaussian blobs are generated and filled with random material textures. These textures contain diverse changes in illumination, scale, rotation, and viewpoint. The generated images are very challenging since they are even visually hard to separate the related regions.

Overall, the contributions in this thesis are major advancements in the direction of solving image analysis problems with Long Short-Term Memory (LSTM) without the need of any extra processing or manually designed steps. We aim at improving the presented framework to achieve the ultimate goal of accurate fine-grained image analysis and human-like understanding of images by machines.

Acknowledgements

I would like to express my gratitude to all those who have supported, influenced and helped me in the process which ultimately resulted in this thesis.

First of all, I would like to thank Prof. Andreas Dengel and apl. Prof. Marcus Eichenberger-Liwicki for their support and encouragement. This dissertation could not be completed without their trust and constructive suggestions. My gratitude also goes to Prof. Thomas Breuel for the opportunity to carry out my research work and his valuable guidance throughout my study. I am also grateful to my committee members, Prof. Didier Stricker and Prof. Katharina Zweig for their valuable comments and suggestions. Furthermore, I would like to thank Prof. Juergen Schmidhuber for the opportunity to visit his group and the great collaboration.

Next, I would like to thank all my colleagues at IUPR and MADM for their fruitful comments and discussions for my research work and this dissertation. In particular, Federico Raue, Mohammad Reza Yousefi, Nibal Nayef, Adnan UI Hasan, and Sebastian Palacio who reviewed my thesis and helped me out with the oral defense. I also thank my formal colleagues, Ludwig Schmidt-Hackenberg, Ilya Mezhirov, Wanlei Zhao, and Muhammet Bastan for the great discussions. I would like to express my appreciation for Ingrid Romani and Brigitte Selzer for their administrative supports. I also want to thank the colleagues at IDSIA for their valuable discussions during my stay, especially Marijn Stollenga for the excellent collaboration. I am very grateful to all other friends, especially Jae-Yeon Chung and Irina Cevallos who helped me out with all the challenges living in Germany and always supported me.

My very special thanks goes to my family. My father Saesang, my mother Insuk, and my brother, Taeyong always supports me during and beyond my thesis. Without them, I could not go through the difficult time during my study. Therefore, I would like to dedicate my thesis to my family.

Contents

Abstract	i
Acknowledgements	iii
List of abbreviations	xii
1 Introduction	1
1.1 Challenges in Image Analysis	2
1.1.1 Challenges in Image Classification	3
1.1.2 Challenges in Image Segmentation	6
1.2 Background of Image Analysis	7
1.3 Research Hypothesis and the Goal of the Thesis	9
1.4 Contributions	10
1.5 Overview of the Thesis	11
2 Image Analysis	14
2.1 Overview	14
2.1.1 Local Feature-Based Approach	15
2.1.2 Learning-Based Approach	21
2.2 Image Classification	24
2.3 Image Segmentation	26
2.4 Conclusion	31

3	Multi-Dimensional LSTM (MD-LSTM) and Its Variant, PyraMiD-LSTM	32
3.1	Background	33
3.1.1	Notations	33
3.1.2	Recurrent Neural Networks	33
3.1.3	Long Short-Term Memory (LSTM)	34
3.2	Traditional Multi-Dimensional LSTM (MD-LSTM)	36
3.3	Proposed PyraMiD-LSTM: Parallel MD-LSTM	38
3.4	Conclusion	41
4	Network Architectures for Image Analysis	42
4.1	Network Layers	42
4.1.1	Input Layer	43
4.1.2	Hidden Layer	44
4.1.3	Output Layers	44
4.2	Network Design for Image Analysis	46
4.2.1	The Dimension of Input	46
4.2.2	Depth of the Network	48
4.2.3	The Type of Output	49
4.3	Network Settings and Generalization	52
4.3.1	Input Representation	53
4.3.2	Weight Initialization	54
4.3.3	Peephole Connections	55
4.3.4	Regularization	55
4.3.5	Optimization	55
4.3.6	Network Parameters	57
4.4	Conclusion	58

5	Image Classification	59
5.1	Texture Classification	60
5.1.1	The Approach	60
5.1.2	Datasets	62
5.1.3	Experimental Setup	63
5.1.4	Results and Analysis	65
5.1.5	Summary	66
5.2	Scene Understanding	67
5.2.1	The Approach	68
5.2.2	Datasets	69
5.2.3	Experimental Setup	74
5.2.4	Results and Analysis	76
5.2.5	Summary	79
5.3	Conclusion	79
6	Image Segmentation	83
6.1	Texture Segmentation	84
6.1.1	The Approach	84
6.1.2	Datasets	85
6.1.3	Experimental Setup	87
6.1.4	Results and Analysis	89
6.1.5	Summary	89
6.2	Scene Labeling	92
6.2.1	The Approach	93
6.2.2	Datasets	93
6.2.3	Experimental Setup	96
6.2.4	Results and Analysis	96

6.2.5	Summary	99
6.3	Conclusion	100
7	Parallel Volumetric LSTM Networks	103
7.1	Biomedical Volumetric Image Segmentation	104
7.1.1	The Approach	104
7.1.2	Datasets	105
7.1.3	Experimental Setup	106
7.1.4	Results and Analysis	108
7.2	Conclusion	110
8	Conclusion and Future Work	113
8.1	Concluding Remarks	114
8.2	Future Directions	119
	Bibliography	120
	Curriculum Vitae	141

List of Figures

1.1	The gap between the visual contents, which are actually present inside an image, and the contents a human focuses on	1
1.2	Examples of tasks and challenges for different image types	3
1.3	Difficulties of texture classification datasets	4
1.4	The difficulties of web-image dataset introduced by this thesis	5
1.5	Difficulties in texture segmentation dataset introduced by this thesis	6
1.6	A stack of EM dataset for 3D volumetric image segmentation	8
2.1	The procedure of local features-based approaches	15
2.2	The Difference of Gaussians (DoG) interest point detector	18
2.3	The Scale-Invariant Feature Transform (SIFT) descriptor	20
2.4	The architecture of CNN	22
2.5	The architecture of CNN	24
2.6	The first post-processing strategy from CNN using super-pixels	27
2.7	The second post-processing strategy from CNN using Conditional Random Fields (CRF)	28
2.8	The third post-processing strategy from CNN using the segmentation tree	29
2.9	The composition of Recurrent Convolutional Neural Networkss (RCNNs)	30
3.1	The architectural difference between RNNs and BRNNs	34
3.2	LSTM memory block	36
3.3	2D-LSTM memory block	38

3.4	Recurrent connections of 2D and Pyramidal Multi-Dimensional LSTM (PyraMiD-LSTM)	39
3.5	Topological difference between MD-LSTM and PyraMiD-LSTM	40
4.1	The internal representation of each layer of the three-layered networks	50
4.2	Various architectures for different applications	52
5.1	A pipeline for texture classification	61
5.2	An overview of the scene analysis system	67
5.3	Image search for database generation	71
5.4	Examples of mid-level attribute data	72
5.5	Examples of scene data for scene analysis and web-image tagging	73
5.6	Confusion table with top-1 predicted semantic attribute on web scene images	78
5.7	The results of automatic web-image tagging	82
6.1	Existing texture segmentation datasets	85
6.2	Blob-Mosaics texture segmentation database	86
6.3	Segmentation results of blob-mosaics images	91
6.4	2D-LSTM network architecture	92
6.5	Visualization of feature maps	94
6.6	The behavior of output activations from the networks while training for scene labeling	95
6.7	Class frequency distribution and confusion table on the Stanford Background dataset	99
6.8	The results of scene labeling on the Stanford Background dataset	101
6.9	Selected mislabeled examples of scene segmentation	102
7.1	Segmentation results for CNNs and PyraMiD-LSTM on EM dataset	111
7.2	Examples of three scan methods used in the Magnetic Resonance (MR) brain dataset and the segmentation results	112

8.1	Examples of mislabeled web-images for tagging.	116
8.2	Examples of failing cases in scene labeling	118

List of Tables

2.1	The architecture of GoogLeNet	26
5.1	Summary of texture datasets used in the experiments	62
5.2	Correct classification rates on five benchmark datasets of texture classification	65
5.3	Accuracy comparisons of single visual attribute classification on web-image dataset	76
5.4	Accuracy comparisons for natural scene analysis on the web-image dataset	77
5.5	The comparison of Mean Average Precision (mAP) on SceneAtt dataset	79
6.1	Accuracy comparison of texture segmentation on texture blob-mosaics images	88
6.2	Pixel and averaged per class accuracy comparison for scene labeling .	97
7.1	Performance comparison on EM images	108
7.2	Performance comparison on MR brain images	109

List of abbreviations

RNN Recurrent Neural Networks

BRNN Bidirectional Recurrent Neural Networks

LSTM Long Short-Term Memory

1D-LSTM One Dimensional LSTM

2D-LSTM Two Dimensional LSTM

3D-LSTM Three Dimensional LSTM

MD-LSTM Multi-Dimensional LSTM

PyraMiD-LSTM Pyramidal Multi-Dimensional LSTM

C-LSTM Convolutional LSTM

CNN Convolutional Neural Networks

RCNN Recurrent Convolutional Neural Networks

DL Deep Learning

ML Machine Learning

CV Computer Vision

NN Neural Networks

MLP Multilayer Perceptron

SVM Support Vector Machines

CRF Conditional Random Fields

RBM Restricted Boltzmann Machines

SOM Self Organizing Map

MR Magnetic Resonance

EM Electron Microscopy

GT Ground-Truth

MAP Mean Average Precision

LoG Laplacian of Gaussian

DoG Difference of Gaussians

MSER Maximally Stable Extremal Regions

SIFT Scale-Invariant Feature Transform

SURF Speeded Up Robust Features

GLOH Gradient Location and Orientation Histogram

HOG Histogram of Oriented Gradients

BoVW Bag-of-Visual-Words

FV Fisher Vector

ReLU Rectified Linear Units

SGD Stochastic Gradient Descent

Chapter 1

Introduction

Given the ever increasing volume of visual data on the Internet, the demand for automated image analysis is growing. However, extracting the desired visual contexts from images of various sizes, qualities, and semantics is a great challenge in the field of Computer Vision (CV). One of the important issues in image analysis is to fill the gap between the visual contents, which are actually present inside an image, and the contents a human focuses on.

Here is an example, the picture of the Eiffel Tower, Figure 1.1. What is the main content of this picture? One viewer may see the Eiffel Tower first, but another viewer may focus on the couple in the front instead of the tower. Others may see the (beautiful) sky, the sunset, or cloud (depending on their prior knowledge or the

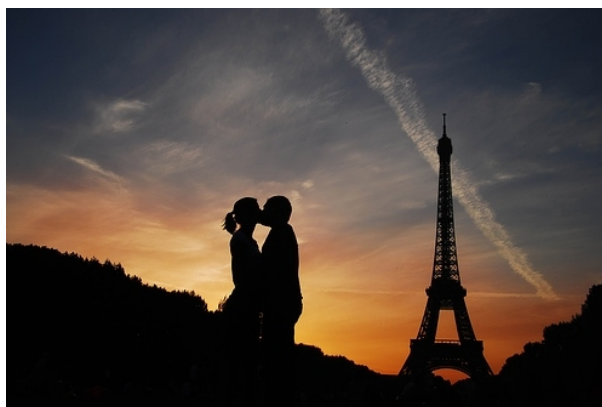


Figure 1.1: This example shows the gap between the visual contents, which are actually present inside an image, and the contents a human focuses on. The picture is taken from <http://favim.com/image/927/>.

information expected from the image). The picture is obtained by querying Google web-search¹ with the string “Eiffel Tower”. However, the top two tags of the image added by users on the website, where the picture is taken from, are “couple” and “dusk”². Which contents should be considered if a machine analyzes this or any other pictures?

Due to the large diversity of images, the different types of images stand for different challenges: finding semantic contents with their location, the overall concept, or the specific contents of the image. Figure 1.2 illustrates some of the tasks and challenges for different image types.

This thesis covers the two CV tasks, of image classification and segmentation, for different types of images (texture images, natural scene images, and 3D biomedical images). In the following section, we discuss the challenges which arise when dealing with such tasks.

1.1 Challenges in Image Analysis

This thesis considers two main tasks of image analysis, image classification and image segmentation. There are numerous related challenges for different types of images considering these tasks. First, image classification is to categorize an image into one or several class labels. An image usually contains a variety of contents³; some of them will be concerned by users, but some may not be. Moreover, there are visually similar images that have different class labels. These issues make the task difficult. Next, image segmentation can be categorized as a pixel-wise classification task. The main issues for this task is to separate the multiple foreground layers, which most people focus on, from the background layer and to label every pixels to the corresponding class labels.

¹<https://www.google.com/imghp>

²<http://favim.com/image/927/>

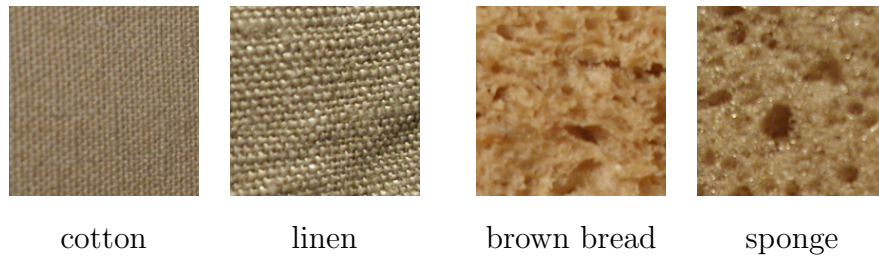
³This kind of image will be referred to as a *complex* image in the remainder of the thesis.



Figure 1.2: Examples of tasks and challenges for different image types

1.1.1 Challenges in Image Classification

For image classification, the thesis takes account of two types of images, texture and scene images. A texture image usually contains a material or a natural texture of an object. A natural scene image is a generic image containing some humans, objects, and background elements. In this section, possible challenges regarding such images will be discussed.



(a) Texture classification dataset, KTH-TIPS [KTH]. “cotton” and “linen” or “brown bread” and “sponge” textures are visually similar.



(b) Texture classification dataset, KTH-TIPS2-a [KTH]. Same texture class in the images are visually different.

Figure 1.3: Difficulties of texture classification datasets. Some different materials may have similar looking textures. On the other hand, the same category materials may look very different. Due to these characteristics, the recognition of textures cannot be easily solved.

Texture images: Texture is a rich source of information about the contents of images and identity of objects. However, reliable texture recognition is challenging because texture is a property of image pixels that is both stochastic and non-local. Most approaches to texture recognition manually design feature extractors to cope with the non-locality, choosing specific ways of integrating information of a region that is robust to changes in phase. An example of such an approach is Haralicks texture features [Har79]. Figure 1.3 shows some challenges for the texture classification task. Texture images might contain different textures in visually similar images



Figure 1.4: The difficulties of the web-image dataset introduced by this thesis. The images contain errors (wrong labels) or noise (logos, watermark, or irrelevant objects).

(Figure 1.3-a) or different-looking images with the same label (Figure 1.3-b). In other words, these contain low inter-class variance (the variance between the classes) and high intra-class variance (the variance within the class), respectively.

Natural scene images: Natural images are often complex and tend to contain much background clutter and/or many unrelated contents. Images obtained from the web may contain a watermark, logo or some text, for instance. Furthermore, images from a social networking service like Twitter or Facebook are often low-resolution. In Figure 1.4, some difficulties of images collected from the web are shown. The focus of the picture also varies depending on the purpose behind taking or posting the picture. In this thesis, a new web-image dataset for more realistic scenarios is generated. In Section 5.2, the difficulties of the natural scene image dataset will be discussed in more detail and along with the discussion on how they can be overcome.

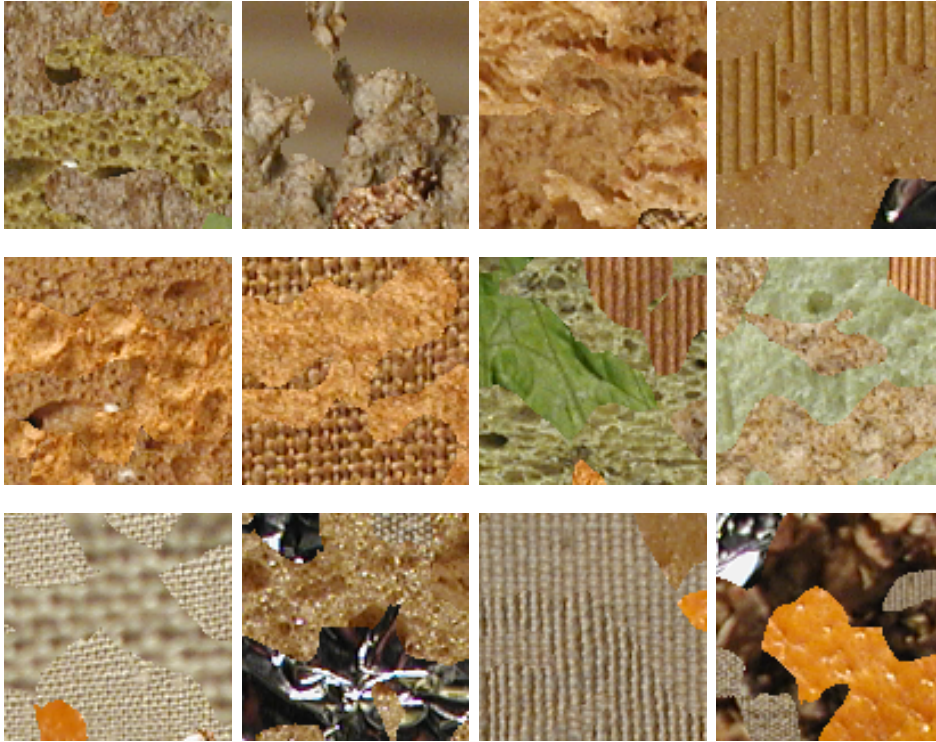


Figure 1.5: Difficulties in texture segmentation introduced by this thesis (Section 6.1.2). The randomly transformed textures (in scale, rotation, and illumination) are positioned in an arbitrary shape. Many of these textures' regions are visually similar or ambiguous.

1.1.2 Challenges in Image Segmentation

Image segmentation is a fundamental task for many applications such as object recognition, medical imaging, and scene analysis. A uniform region is defined by homogeneous material or between discontinuities in depth. Here, the various challenges of dealing with texture images, natural scene images, and 3D biomedical images will be discussed.

Texture images: The texture has major visual cues of the surface within and between the regions. In order to segment the disjoint uniform regions based on textures, the combination of texture classification and image segmentation is used. However, it is difficult to generalize the system in order to find a pattern without the knowledge of domain, as it is affected by various external conditions, i.e., wide range of scale, illumination, rotation, as well as internal noise of the texture. Figure 1.5 shows some examples of such challenges on an image segmentation dataset.

Natural scene images: Accurate image segmentation on scene images (i.e., scene labeling) is an important step towards image understanding. The scene labeling task consists of partitioning the meaningful regions of an image and labeling pixels with their regions. Pixel labels can (most likely) not only be decided by low-level features, such as color or texture, extracted from a small window around pixels. One challenge when dealing with such a setting is to distinguish “grass” from “tree” or “forest”. As a matter of fact, human people perceptually distinguish regions via the spatial dependencies between them. For instance, visually similar regions can be predicted as “sky” or “ocean” depending on whether they are on the top or bottom part of a scene.

3D biomedical images: Analyzing biomedical images is one of the most important subjects of study for biologists, but especially for neuroanatomists. There are many biomedical 3D volumetric data sources, such as Computed Tomography (CT), MR, and Electron Microscopy (EM). This volumetric image data provide higher dimensional information but is hard to handle. Moreover, a lot of noise and low quality images make the task very challenging. Many researchers in this field have been investigating reliable automated segmentation and reconstruction of this data. One of the common solutions is to process each 2D slice separately, using image segmentation algorithms such as snakes [KWT88], random forests [WGS⁺15], and CNNs [CGGS12]. Nevertheless the slices of a volume are continuous, the contextual information between slices cannot be contributed to the final classification of a pixel with such approaches. In other words, there is no easy way to integrate the full context of each pixel in such a volume. Figure 1.6 shows some examples of 3D biomedical volumetric datasets.

1.2 Background of Image Analysis

Designing a system, which is capable of solving the challenges discussed above, is one of the main goals in CV communities. In both classification and segmentation, the typical approaches are to find an appropriate representation which the system attempts to analyze. In such approaches, the important issue is to select not only the discriminative representation between the class, but also the generalized representation within the class which can be applied to a large variety of data.

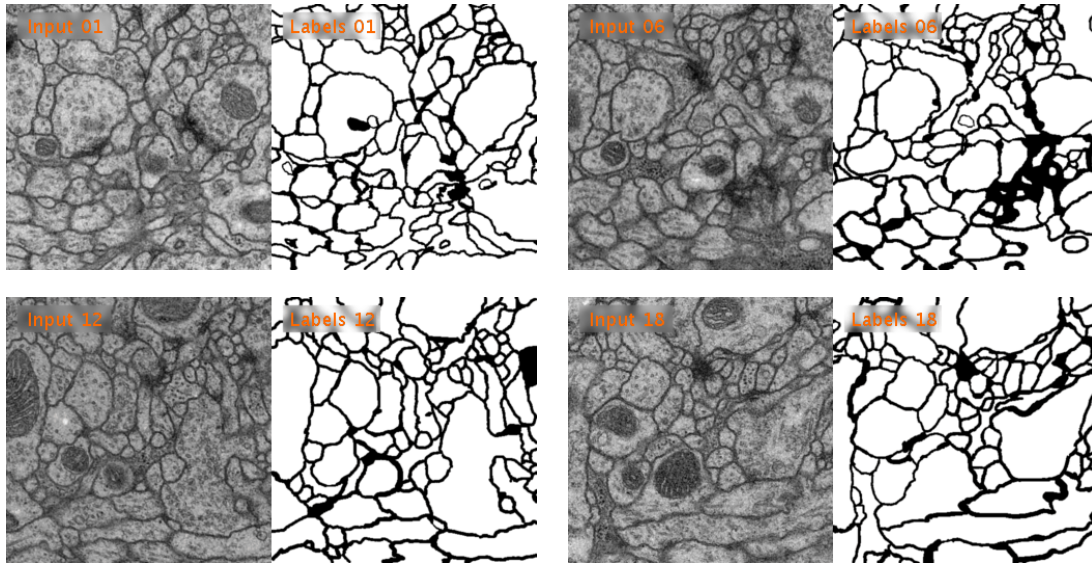


Figure 1.6: A stack of EM dataset [Seg12] (slice 1, 6, 12, and 18 from 30 slices). To generate such a dataset, the electron beam travels in straight lines and images are captured in (short) consecutive time points. Therefore, the slices are closely linked.

Local interest point detectors and descriptors have been proven for many years to provide a robust and highly adaptable way to represent images. Typically, an interest point detector selects distinctive points or regions of an image, and a descriptor characterizes these regions using color, texture, shape, location, and so on. The main question arising from these in such approaches is whether or not the extracted information is invariant to the common image transformation (e.g., scaling, rotation, or translation). Various detectors based on the edge⁴, corner⁵, and blob⁶ and the descriptors⁷ have been introduced in the literature. More details will be discussed in Section 1.5.

Another direction, which has been studied, is to emulate the behavior of the human brain and visual system, providing more biologically plausible alternative to statistical learning methods. Some examples of these biologically inspired models are Neocognitron [Fuk80], Convolutional Neural Networks (CNN) [LBBH98a], and Long Short-Term Memory (LSTM) [HS97b]. More recently, this research field has been extensively growing as an end-to-end vision system based on Deep

⁴e.g., Canny [Can86], Deriche [Der87]

⁵e.g., Harris [HS88]

⁶e.g., Maximally Stable Extremal Regions (MSER) [MCUP04]

⁷e.g., SIFT [Low99], Speeded Up Robust Features (SURF) [BETG08], Gradient Location and Orientation Histogram (GLOH) [MS05b], Histogram of Oriented Gradients (HOG) [DT05]

Learning (DL) [Ben09, Sch14, DY14]. The related algorithms (especially CNNs) have become very successful in many computer vision applications such as image classification [CMS12, KTS⁺14], traffic sign recognition [SL11], image caption generation [VTBE14, KSH12b], video classification [KSZ14], and face recognition [LGTB97].

1.3 Research Hypothesis and the Goal of the Thesis

The aim of this thesis is to undertake CV tasks in an efficient manner. Similar to other DL approaches, the end-to-end system is accomplished solving the various image analysis tasks mentioned above. One of the popular methods in DL is CNN which handle only small local context of the pixels to be classified. Unlike CNN, given its structural nature, LSTM, containing some internal memory storage, is able to cooperate with the local (pixel-by-pixel) and global (label-by-label) dependencies in a single process. Although CNNs for image data have been rewarding, LSTM-based methods are not explored yet in CV. The main focus of this thesis is to expand the study in both DL and CV especially with LSTM.

The research hypothesis of this study is that:

Recurrent Neural Networks (RNN) with LSTM represent a general and powerful sequence learning method. Especially, Multi-Dimensional LSTM (MD-LSTM) networks, due to the architectural nature, should be able to achieve reliable and accurate image analysis by integrating entire spatial and/or temporal context of a pixel. Therefore, the system not only handles for 2D images but also higher dimensional data (3D or 4D), and resolves the challenges mentioned above. It leads to a comprehensive end-to-end vision system, which requires local and global contextual information of each pixel to be predicted, using raw pixel values rather than selected features on complex real-world input data.

To test this hypothesis and demonstrate the feasibility of a high performance LSTM-based visual system, the contributions of this thesis will be presented in the following section.

1.4 Contributions

Based on the challenges discussed in the previous section, this thesis stands for the following contributions. The contributions are concerned mainly in the field of DL and CV.

In Deep Learning

D-1: (**Chapter 3**) The highly promising architecture of MD-LSTM has a limitation; previous MD-LSTM implementations could not exploit the parallelism of modern GPU hardware. Therefore, the traditional MD-LSTM cannot easily be applied to a high-dimensional and a big data input. Here, to cope with 3D volumetric data, the traditional MD-LSTM is re-designed to Pyramidal Multi-Dimensional LSTM (PyraMiD-LSTM). With a different topology and update strategy, PyraMiD-LSTM is easier to parallelize, needs fewer computations overall, and scale well on GPU architectures.

D-2: (**Chapter 3 and 4**) Two LSTM models for higher dimensions, the traditional MD-LSTM and the proposed PyraMiD-LSTM, and their network architectures are explored for image-wise and pixel-wise classification problems. The single-layer and the deep network architectures (sometimes combined with additional layers) resolve various image analysis tasks on several types of images. Here, various network design choices for image analysis are suggested. The design choices are mainly based on three factors: (1) the dimension of input, (2) depth of the network, and (3) the type of output. All possible architectures are analyzed and evaluated. Furthermore, the major network settings and generalization techniques are evaluated, particularly for the LSTM network.

In Computer Vision

C-1: (**Chapter 5 and 6**) 2D-LSTM networks resolve diverse CV problems without additional processing, e.g., pre-/post-processing or manual feature extraction. The problems of texture image classification and segmentation, scene understanding and labeling tasks are addressed in the thesis. The approach yields performance gain compared to other methods including CNNs for all the tasks mentioned above, yet using a simpler model and much fewer parameters.

- C-2: (**Chapter 7**) PyraMiD-LSTM addresses the problem of combining spatio-temporal context of each pixel on 3D volumetric images. The entire volume is processed in a single network which takes advantage of these full contextual information of 3D segmentation. PyraMiD-LSTM achieved the best pixel-wise image segmentation results in the MR brain image segmentation contest [A. 15]⁸ and competitive results on EM images [Seg12] without post-processing⁹.
- C-3: (**Chapter 6**) Automated texture segmentation dataset generation is proposed to overcome the limitations of existing texture segmentation datasets. It creates diverse texture blob-mosaics with their corresponding Ground-Truth (GT). All generated texture blob-mosaics are randomly shaped and consider the following transformations: scale, rotation, and illumination. Furthermore, a corresponding evaluation scheme to measure the performance with state-of-the-art algorithms is described.
- C-4: (**Chapter 5**) A new web-scene image dataset is introduced for a realistic scene understanding system. Single-attribute training data for natural scenes (a single label per image) make the training step easier than having full images with multiple attributes for complex web-scene image analysis. With the help of the proposed pruning strategies, a complete scene understanding system classifying multiple labels has been accomplished. Furthermore, automatic web-image tagging is illustrated using the dataset as a more realistic application.

1.5 Overview of the Thesis

The goal of this thesis is structured into five main chapters. Chapter 2 describes the background of image analysis. Chapter 3 explains different LSTM models for 2D and 3D data for different tasks. The remainder of chapters introduces the details of each task with a different LSTM network architecture. Chapter 5 and Chapter 6 present the details of image classification and segmentation tasks with MD-LSTM networks, respectively. Finally in Chapter 7, PyraMiD-LSTM is proposed for 3D volumetric image segmentation, before the thesis is concluded in Chapter 8.

⁸The results can be found in <http://mrbrains13.isi.uu.nl/results.php>

⁹The results can be found in http://brainiac2.mit.edu/isbi_challenge/leaders-board

In Chapter 2, a brief overview of image analysis tasks is given. Additionally, the related works on image classification and segmentation are also summarized for better understanding of following chapters.

In Chapter 3, LSTM-based methodologies for image analysis tasks are described. First, the main structure of standard RNNs and Bidirectional Recurrent Neural Networks (BRNNs) are compared. The study of LSTM and MD-LSTM is then described in more detail which is followed by a new sophisticated LSTM, PyraMiD-LSTM to resolve some limitations that the traditional MD-LSTM has.

In Chapter 4, the details of all possible layers are described before suggesting the possible network architectures for image analysis tasks on various types of images. Furthermore, the specific network settings and generalization techniques for the network training is explained.

In Chapter 5, the focus lies on *classification* tasks of texture and natural scene images: texture image classification and scene understanding. Different learning strategies for texture classification are explored. Diverse input and output representation schemes dramatically increase the performance. These schemes are then extended to the web scene images which contain a wide variety of noise (e.g., watermarks and unrelated contents). To handle this issue, several pruning rules are introduced (to retain the system robust). Also a new web-image dataset is created (to have a robust training). These strategies are successfully adapted to a web-image tagging system and show the performance with a large margin compared to other baseline approaches including CNNs. Furthermore, the system is also evaluated on the publicly available dataset (outdoor scene attribute dataset) to show the performance gain (about 21%).

In Chapter 6, we move our focus to the *segmentation* tasks on texture and natural scene images: texture image segmentation and scene labeling. First, a new Automated Texture Blob-Mosaics Database Generator is proposed to overcome the limitations of current texture segmentation datasets. Here, a new evaluation criteria is also proposed. The evaluation on this dataset for the segmentation task shows that 2D-LSTM network architecture with only a single layer performs better than

other texture image-based segmentation algorithms. Secondly, a deeper model of 2D-LSTM networks for segmentation is studied to carry more complex images out; natural scene images may contain a lot more clutter and noise. The networks take into account the complex spatial dependencies of each pixel and accomplish classification, segmentation, and context integration all within one model. With much lower computational complexity compared to other DL methods, the LSTM model achieves state-of-the-art performance over two popular scene labeling datasets.

In Chapter 7, a new LSTM model, PyraMiD-LSTM, is proposed. The context information flow is re-arranged from cuboid (in MD-LSTM) to pyramidal for better parallelization especially for 3D volumetric data. This model is applied to two challenging datasets involving segmentation of biological volumetric images and achieve competitive or the best results on two datasets.

In Chapter 8, methodologies, contributions, and results are summarized. Further possible challenges are discussed as future work.

Chapter 2

Image Analysis

This chapter provides a background on image analysis, with a focus on image classification and segmentation. It outlines the recent methodologies that are connected to the work presented in this thesis.

Image analysis consists in understanding the contents of given images by detecting objects or faces, removing noise, detecting shapes or edges, extracting regions or visual contents. The analysis of images has been extensively studied for several decades. There are two different directions of approaches for image analysis: local feature-based and learning-based approach. The first direction had been popular in the CV field in early years; they are also called early vision algorithms. More recently, the second direction has been densely studied and improved. As mentioned in Chapter 1, there are many challenges in achieving reliability and accuracy in an image analysis system. This chapter describes basic ideas to solve the issues and a brief overview of the most common approaches used in this field, leading to the motivation of the study for the remaining chapters.

Section 2.1 provides an overview about the typical approaches of image representation for image analysis. Section 2.2 and Section 2.3 cover the key state-of-the-art methods, especially for image classification and segmentation.

2.1 Overview

Approaches for image analysis are divided into two major directions.

- Handcrafted local features: Manually designing the specific feature representation according to the task and type of images
- Learning-based features: Automatically learning feature representation from raw input data

This section provides an overview about commonly used methods in each direction.

2.1.1 Local Feature-Based Approach

The main procedure of this direction is illustrated in Figure 2.1. The most distinctive points/regions (local interest point/region detectors) are first selected, then the most relevant information around the points/regions are extracted for the specific task (local descriptors). In the end, a classifier is utilized for final image- or pixel-level classification over the extracted descriptions.

Feature Detection

There are many ways of detecting and representing relevant regions based on parts, intensities, gradients, color, texture, or mixtures of them. The local detector selects a set of (local) interest points or regions which are extracted from stable, reliable, and unique positions on different images. Most of detectors find the local area in the image with a large variation in intensity (edges) in all directions (corners). The field of this research has a long history, but the commonly used local detectors are Hessian [Bea78], Harris [HS88], and Laplacian [Lin98].

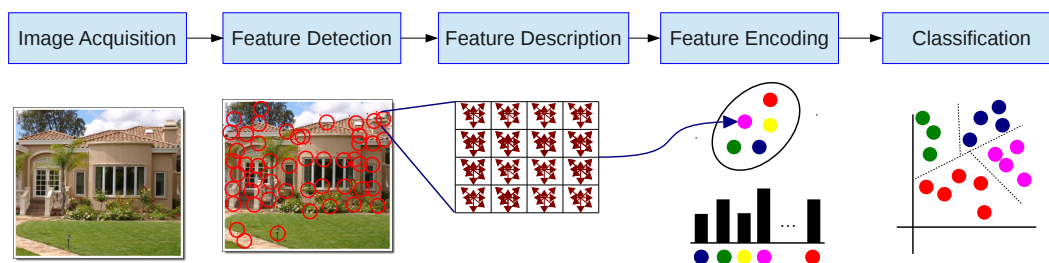


Figure 2.1: The procedure of local features-based approaches. The typical pipeline is composed of the four steps: (1) feature detection (e.g., key-point detection), (2) feature description (e.g., SIFT), (3) Feature encoding (e.g., BoVW), and (4) classification (e.g., SVM). More details of each step will be discussed in the following.

Hessian detector: The Hessian detector [Bea78] searches for an image point which has strong derivatives in two orthogonal directions using the matrix of second order derivatives.

$$H(x, y) = \begin{bmatrix} I_{xx}(x, y) & I_{xy}(x, y) \\ I_{xy}(x, y) & I_{yy}(x, y) \end{bmatrix}, \quad (2.1)$$

where I_{xx} , I_{xy} , and I_{yy} are the second partial derivatives in the x direction, in the x and the y direction, and in the y direction respectively. The maximum of the determinant of the Hessian matrix becomes an interest point.

$$\det(H) = I_{xx}I_{yy} - I_{xy}^2 \quad (2.2)$$

Harris corner detector: Harris [HS88] corner is detected based on the changes in image intensity around a point (x, y) using the second momentum matrix.

$$C(x, y) = \begin{bmatrix} I_x^2(x, y) & I_x I_y(x, y) \\ I_x I_y(x, y) & I_y^2(x, y) \end{bmatrix}, \quad (2.3)$$

where I_x and I_y are the first derivatives in x and y directions respectively. In general, Gaussian filters are used to calculate the image derivatives. The corner is defined when the first two eigenvalues are both large, otherwise, if only one of them is large, the point is defined as an edge. Instead of explicitly comparing eigenvalues, the determinant and the trace of the matrix $C(x, y)$ can be computed to find the point. The interest point should satisfy the following condition:

$$\det(C) - \alpha(\text{trace}(C))^2 = \lambda_1 \lambda_2 - \alpha(\lambda_1 + \lambda_2)^2 > t, \quad (2.4)$$

where α is a constant [0.04 – 0.06] and t is a threshold.

Since these approaches have their own distinct characteristics, the combination of those key-point detectors, as well as the addition of other invariance properties like scale or rotation are investigated. Examples of these approaches are Harris-Laplace [MS02], scale invariant Harris-Laplacian [MS04], Laplacian of Gaussian (LoG) [Lin98], and Difference of Gaussians (DoG) [Low04]. The detectors explained above are robust to rotation, illumination changes, and noise, but not to changes of a scale. To deal with scale invariance, LoG and DoG detectors are introduced using a scale space representation. A scale space representation is the most widely used multi-

scale representation, based on the scale space theory introduced by Witkin [Wit84]. The scale space is a set of Gaussian smoothed images of the original image with various sizes of kernels. It is subsequently combined with the pyramid representation, in which each scale is represented by various image resolutions.

Laplacian of Gaussian (LoG): The LoG [Lin98] detector can be computed based on the scale space representation.

First, the Gaussian kernel is computed as follows:

$$G(x, y) = \frac{1}{2\pi\sigma_n^2} e^{-\frac{x^2+y^2}{2\sigma_n^2}}, \quad (2.5)$$

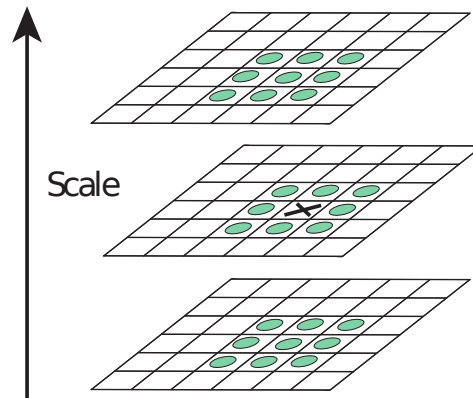
where n is the scale. LoG is then computed by the second derivatives of the Gaussian $\nabla^2 G(x, y)$ at each scale n . Finally, the interest regions are detected based on the maximal response of their neighbors over all scales.

Difference of Gaussians (DoG): Instead of all computations of LoG in the scale space, this process can be approximated by DoG, which is computed by subtracting two adjacent scales.

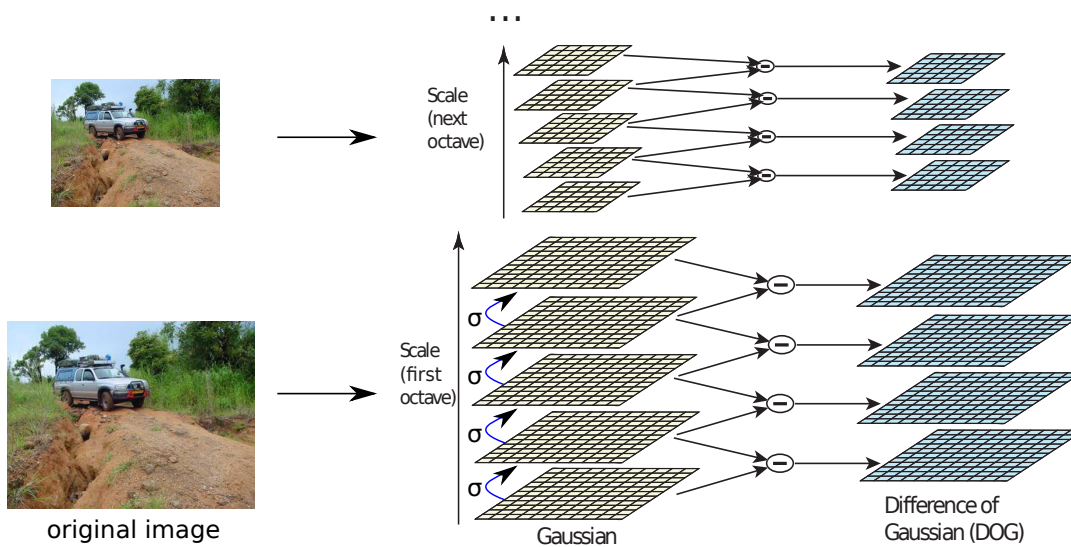
$$D(x, y) = G(x, y; \sigma_{n+1}) - G(x, y; \sigma_n) \quad (2.6)$$

The interest points are found by comparing the pixel's eight neighbors and with the nine neighbors of the adjacent level scales (see Figure 2.2-a). DoG is generally combined with the pyramid representation; several scales n are computed on various image resolutions (see Figure 2.2-b). This increases the accuracy of detecting robust interest point locations.

Harris-Laplacian interest point detector: The Harris-Laplacian detector [MS04] builds up two separate scale spaces for the Harris function and the Laplacian. The interest points are selected based on localized candidate points on each scale (Harris) and the maximal response over scales (Laplacian). The detected points are robust to scale changes, rotation, illumination changes, and noise.



(a) The point selection in a scale space



(b) The DoG interest point detector

Figure 2.2: The DoG interest point detector. (a) The interest point is selected from the maxima response of the DoG. The current pixel (marked as X) is compared to the 8 neighboring pixels of the current scale and the 18 neighboring pixels of the adjacent level scales. (b) The DoG is performed in the pyramid scale space. The difference between the Gaussian smoothed images with different σ is computed (a scale space). This process is performed in several image resolutions (an image pyramid). The figure is reproduced from [Low04].

Feature Description

The most popular descriptors used with the detectors mentioned above are gradient-based descriptors that use local histograms of image gradients. For example, SIFT [Low99], HOG [DT05], shape contexts [BMP02], and generalized shape contexts [MM03] compute orientation and/or spatial histograms of the local region around a pixel. These approaches themselves take scale and rotation invariance into account. Here, the two most popular descriptors, SIFT and HOG will be explained in more detail.

Scale-Invariant Feature Transform (SIFT): In general, SIFT [Low99] is combined with the DoG interest point detector. The main focus of this descriptor is to achieve the robustness to deformations, noise and small translation (shifting). The localized gradient orientation histogram is constructed on each detected point. The image gradient magnitude $m(x, y)$ and the orientation $\theta(x, y)$ are first computed around the detected points (a 16×16 grid).

The computations are in the following:

$$m(x, y) = \sqrt{(G(x+1, y) - G(x-1, y))^2 + (G(x, y+1) - G(x, y-1))^2}, \quad (2.7)$$

$$\theta(x, y) = \text{atan2}(G(x, y+1) - G(x, y-1), G(x+1, y) - G(x-1, y)), \quad (2.8)$$

where $G(x, y)$ is the Gaussian smoothed image. The orientation histogram is created from 4×4 subregions with 8 bins each in a 16×16 grid (see Figure 2.3). When accumulating the histogram from the orientation $\theta(x, y)$, each bin is weighted by the magnitude $m(x, y)$. Therefore, $4 \times 4 = 16$ histograms with 8 bins create a vector with 128 elements, which becomes a vector of the feature descriptor for an interest point. This vector is then normalized to the unit length. This normalization adjusts the changes of image contrast. The values are then thresholded with a value of 0.2 and normalized again to the unit vector in order to reduce the non-linear illumination changes. There are extensions like color SIFT and dense SIFT that compute SIFT on the color space and on the whole image (without any detector), respectively.

Histogram of Oriented Gradients (HOG): HOG [DT05] computes a gradient orientation histogram as SIFT. The difference from SIFT is that the computation is performed over a grid of an entire image; the SIFT descriptor is computed around

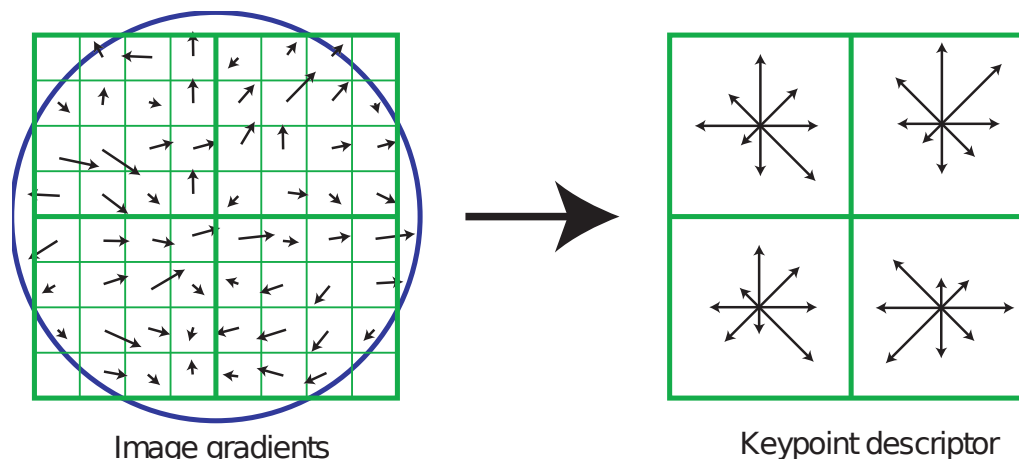


Figure 2.3: The SIFT descriptor. The gradient magnitude and orientation are computed around 16×16 window of the detected interest point (left). The orientation histogram is then constructed over 4×4 subregions (right). In this figure, 2×2 subregions from 8×8 are shown. The figure is reproduced from [Low04].

a scale-invariant interest point. In addition, each bin of the histogram in HOG represents the number of edge pixels having orientations. The HOG descriptor is normalized with respect to image contrast, but not with respect to orientation. This descriptor was especially successful with human detection [DT05].

Feature Encoding

Besides, one of the most successful frameworks based on SIFT-like features is BoVW [AR02, OFPA04, CDF⁺04, MS05a], which encodes the descriptions into code-books. The similar descriptions are first grouped and clustered together, then the center of each group is defined as a representative of the group, building a visual code-book. When classifying the images, each descriptor finds the closest visual words and the category with the most frequent words is selected.

Classification

At the end, these representative features are classified using a classifier. One of the most widely used classifiers is the Support Vector Machines (SVM) with a pre-defined kernel such as linear or Chi-square. The SVM classifier has been combined with the descriptors mentioned above and applied in diverse CV applications such as object detection [PP00], object recognition [DS03], image classification [DBLFF10], human action classification [NFF07], and human body detection [RST02]. More recently, Random Forests classifier has become an alternative classifier in numerous

tasks such as object segmentation [SCZ08], image classification [BZM07a], and object detection [GRVG12].

These approaches generate a very compact representation of an image, and different methods can be easily combined to provide additional robustness. Despite the success of these approaches, the main drawback of this framework is that the performance depends heavily on the feature representation of the input. However, the manually designed feature representation is usually task-specific and both over-specified and incomplete.

2.1.2 Learning-Based Approach

The second direction comes from Deep Learning (DL)¹ [Ben09, DY14, Sch14], in the field of Machine Learning (ML), which tackles the drawback mentioned above. DL-based approaches learn or try to model the high-level representation of input data by using multiple layers with some non-linear operations. In general, these learning-based models require several important components [BL07]: the representation of the data (pre-processing or feature extraction), the architecture of a model (types of layers and operations), the loss function, and the regularizer.

Several factors have been discussed by Bengio et al. [BCV13]² to explain why DL becomes an outstanding direction. First, it provides automated feature learning, which resolves the drawback of the previous approaches mentioned above. Secondly, it provides a distribute representation. When input is huge and sparse, the model from learning-based algorithms can easily overfit. However, DL-based approaches can learn a large amount of features in different levels, which provide diverse and sparse representation from the large input. When the network gets deeper, it can learn different levels of abstraction of images — gradually more abstract in higher layers. Therefore, the system can provide automated feature learning with general image representation without any prior knowledge of the task.

Previously, deep but shallow NNs have been applied to image analysis with some engineered features. In general, a deep model is hard to train as it can easily lead to local optima [EBC⁺10]. To avoid this issue, a weight initialization technique of NN as pre-training using Restricted Boltzmann Machines (RBM) is introduced [HS06,

¹Note that here the focus of Deep Learning (DL) lies in Neural Networks (NN).

²The authors referred to it as “representation learning”

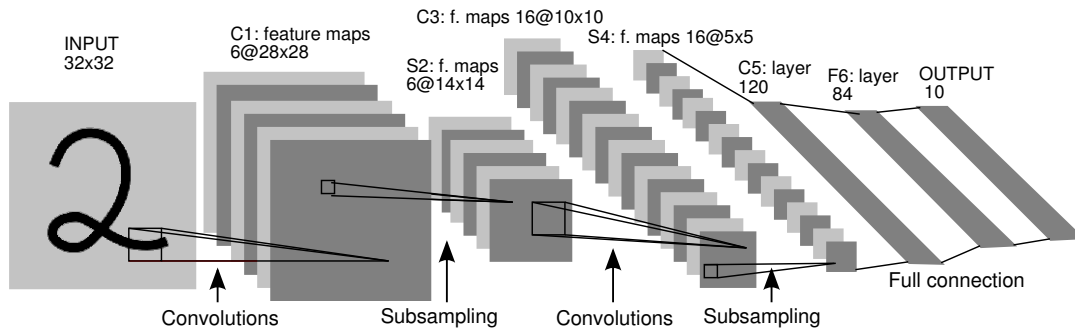


Figure 2.4: The architecture of CNN introduced by LeCun et al. [LBBH98a]. The network consists of convolutional layers (convolutions), pooling layers (subsampling), and several fully connected layers. The network outputs ten classes of handwriting digits. The ReLU layer is not included in this architecture. Each plain represents a feature map. The figure is reproduced from [LBBH98a].

EBC⁺10]. The new breakthrough in CV is to train very large and deep CNN on the ImageNet database³ [KSH12b], which contain huge and diverse natural images used for image classification. Therefore, the DL-based end-to-end system became one of the most common approaches for image analysis.

Convolutional Neural Networks (CNN)

One of the most popular approaches, CNNs [Fuk80, LBBH98b], is a variation of Multilayer Perceptron (MLP) designed to reduce the pre-processing steps that are commonly necessary for image analysis tasks. In CNNs, each pixel is processed by a single neuron, which acts like a filter (called a “filter bank”), and these filters are learned from the input (unlike other filter-based feature extractors). The filters are shared over all pixels (weight sharing) which reduces the number of weights and the size of the networks. The network contains various types of layers: an input layer, convolutional layer(s), Rectified Linear Units (ReLU) layer(s), pooling layer(s), and fully connected layer(s) with a loss function. Figure 2.4 shows one of the CNN architecture introduced by LeCun et al. [LBBH98a].

The input layer first takes a raw pixel value of the image.

The convolutional layer then computes a rectangular grid of neurons, which become a filter or a feature map. Given a set of filters (weights of neurons)

³<http://image-net.org/>

$w_1, w_2, \dots, w_k, \dots, w_K$, where K is the number of neurons of the layer,

$$f_k = w_k * x, \quad (2.9)$$

where x is an input image or output from the previous layer, $(*)$ is a convolution operation, and f_k is the k th feature map. This layer results in feature maps (filters) with the size $n \times n \times q$, where n is smaller than the dimension of the image, and q is the number of channels (3 in color). These filters are locally connected, and each of them is convolved with the image to produce K feature maps that learn the variety of features from the same image.

The ReLU layer computes an element-wise activation function to increase nonlinearity:

$$f(x) = \max(0, x) \quad (2.10)$$

Compared to other commonly used activations functions like sigmoid, or hyperbolic tangent, ReLU is faster without affecting the size of dimensions ($n \times n \times k$).

The pooling layer takes small rectangular blocks from the ReLU layer that are sub-sampled in this layer. Typically mean or max-pooling over $p \times p$ regions is performed, where p is between 2 to 5. A max-pooling function can be computed as:

$$P_V = \max_{v \in V} h_v, \quad (2.11)$$

where v is the index of the pooling region (within the $p \times p$ region), V is the pooling region, and h is an activation within the region. This layer helps to create translation-invariant features. After the pooling layer, another nonlinearity function like additive bias or sigmoid is applied to each feature map.

The fully connected layer computes the final class score after several stacks of convolutional, ReLU, and pooling layers. The fully connected layer is the same as MLP (with weights W and biases b). The network is typically trained with back-propagation and some loss function such as the cross-entropy function with the softmax function (the details of the loss function will be discussed in Chapter 4).

The research on CNN has been rapidly progressing in recent years for CV tasks such as image classification [KSH12b], object recognition [GDDM14a], object detection [TGJ⁺15], action recognition [SZ14, JXYY13], video classification [KTS⁺14], pose estimation [PSCZ15], face detection [LLS⁺15], and scene labeling [LSD15, PC14,

FCNL13, KEF⁺14].

The following sections focus on more specific tasks of image analysis, image classification and segmentation.

2.2 Image Classification

Image classification is the task of assigning a given image to one of many predefined class labels. The label can be an object or other visual concept of the image. Previously, commonly used databases in this research (e.g., CIFAR⁴, MNIST⁵) mainly dealt with a small number of classes and low-resolution images. In traditional CV, the approaches are often confined to these restricted databases. As image classification data resources become larger and more complex (e.g., ImageNet⁶), a generalization of such tasks becomes more important. Moreover, current digital images found on the web are rich in content and tend to have target objects occupying a large fraction of the input image. Thus, not all of the contents are related to the class label of the input image. Also, the information the contents carry is in general very redundant.

Earlier research focused more on separating the foreground from the background or extracting informative features based on plain visual cues like texture, color, and shape. The former approaches conducted segmentation first as a pre-processing step to discard background clutters or unrelated contents. The performance of segmentation is very important in this case, as the output of the segmentation is used for the classification. The issues of segmentation will be discussed in the next section. The latter approaches use sophisticated handcrafted feature detectors and descriptors, sometimes combined with simple classifiers discussed in the last section.

Recently in DL, most of the cited works for image classification make use of CNNs. In fact, CNNs became very popular in this field since Krizhevsky et al. [KSH12a] won the ImageNet challenge 2012 for the first time with CNNs⁷. They applied CNNs to the biggest database for image classification, ImageNet, containing 1000 categories and 1.2 million images. To train a large amount of data, the networks have become wide and deep; five convolutional layers (with pooling), three fully-connected layers,

⁴<http://www.cs.toronto.edu/~kriz/cifar.html>

⁵<http://yann.lecun.com/exdb/mnist/>

⁶<http://image-net.org/>

⁷<http://image-net.org/challenges/LSVRC/2012/results>

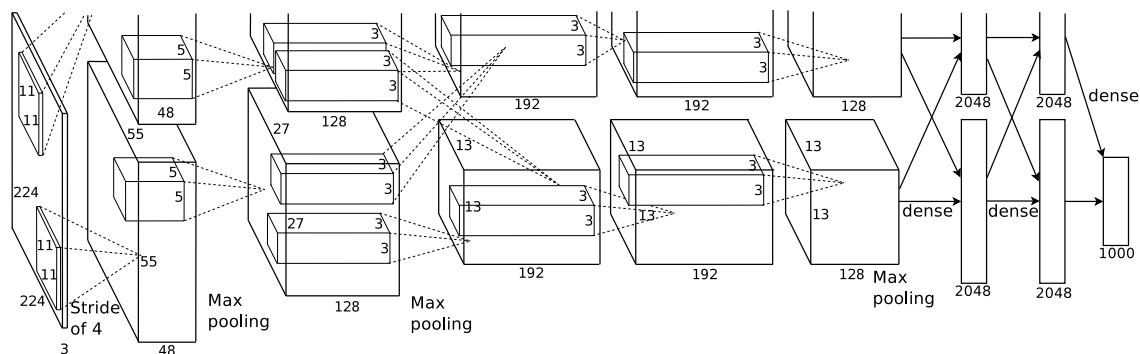


Figure 2.5: The architecture of CNN. The network consists of convolutional layers (convolutions), pooling layers (subsampling), and several fully connected layers. The network outputs ten classes of handwriting digits. The ReLU layer is included in the later architectures of CNN. Each plain is a feature map. The figure is reproduced from [KSH12a].

as well as a ReLU layer explained earlier (in Section 2.1) are included. The overall architecture is shown in Figure 2.5. The input layer receives the input image with the size $224 \times 224 \times 3$. The first convolutional layer filters 96 kernels of size $11 \times 11 \times 3$ with a stride of 4 and is followed by max-pooling. The second convolutional layer has 256 kernels of size $5 \times 5 \times 48$, then the outputs are max-pooled. The third convolutional layer has 384 kernels of size $3 \times 3 \times 256$, the fourth has 384 kernels of size $3 \times 3 \times 192$, and the fifth has 256 kernels of size $3 \times 3 \times 192$. The last three convolutional layers do not combine with the pooling layer. Finally, two fully connected layers with 4096 neurons are presented before the output layer with 1000 classes. The ReLU layer is applied to every convolutional and fully connected layers. Since both the size of the networks and the amount of data are huge, the computations are extremely expensive (60 million parameters and 650,000 neurons). Therefore, an efficient GPU implementation (with two GPUs) was introduced. The authors also introduced some techniques to avoid overfitting such as data augmentation and dropout; these techniques will be explained in Section 4.3.

Later, research shifted its focus on increasing the number of layers (deeper) and the size of hidden units per layer (wider) [LCY13, SEZ⁺13, ZF14, SLJ⁺15]. One of the models by Szegedy [SLJ⁺15], known as GoogLeNet, won the same challenge as Krizhevsky's in 2014⁸. They carefully designed the depth and width of the network by stacking the local optimal sparse structure, hence being able to train a 22 layer deep model. The main architecture of GoogLeNet is summarized in Table 2.1.

⁸<http://image-net.org/challenges/LSVRC/2014/results>

Table 2.1: The architecture of GoogLeNet. The network has 22 layers (layers with parameters; or 27 layers including pooling layers). An inception layer is a sub-network combining several convolution filters with 1×1 reduction filter and a max-pooling layer. Here, nine inception layers are added with the width (the size of layer) range from 256 to 1024 filters. In the first row, ‘params’ is the number of parameters, and ‘ops’ is the number of operations. The table is reproduced from [SLJ⁺15].

type	patch size (stride)	output size	depth	params	ops
convolution	7×7 (2)	$112\times 112\times 64$	1	2.7K	34M
max pool	3×3 (2)	$56\times 56\times 64$	0		
convolution	3×3 (1)	$56\times 56\times 192$	2	112K	360M
max pool	3×3 (2)	$28\times 28\times 192$	0		
inception		$28\times 28\times 256$	2	159K	128M
inception		$28\times 28\times 480$	2	380K	304M
max pool	3×3 (2)	$14\times 14\times 480$	0		
inception		$14\times 14\times 512$	2	364K	73M
inception		$14\times 14\times 512$	2	437K	88M
inception		$14\times 14\times 512$	2	463K	100M
inception		$14\times 14\times 528$	2	580K	119M
inception		$14\times 14\times 832$	2	840K	170M
max pool	3×3 (2)	$7\times 7\times 832$	0		
inception		$7\times 7\times 832$	2	1072K	54M
inception		$7\times 7\times 1024$	2	1388K	71M
avg pool	7×7 (1)	$1\times 1\times 1024$	0		
dropout (40%)		$1\times 1\times 1024$	0		
linear		$1\times 1\times 1000$	1	1000K	1M
softmax		$1\times 1\times 1000$	0		

The inception layer is a set of convolutional layers and a pooling layer. This layer concatenates several sizes of convolution filters (1×1 , 3×3 , and 5×5) with the max-pooling (3×3). In addition, 1×1 convolutions with ReLU are used before 3×3 and 5×5 convolutions and after max-pooling to avoid a computational blow up within a few stages. These 1×1 convolutions keep the representation sparse and compress the information efficiently. Overall, the number of layers including all blocks in the inception layers is around 100.

2.3 Image Segmentation

Accurate image segmentation is an important step towards image understanding. The image segmentation task consists of partitioning the meaningful regions of an image and labeling pixels with their regions. The higher-level representation of images (their global context) is typically constructed based on the similarity of the low-level features of pixels and on their spatial dependencies using a graphical model. The graphical models construct the global dependencies based on the similarities of neighboring segments. The most popular graph-based approaches are Markov Random Fields (MRF) [GFK09a, KK10, LJ08, TL10] and Conditional Random Fields (CRF) [HZCP04, RTK09]. However, such methods require pre-segmentation, super-pixels, or candidate areas.

More recently, DL has become a very active area of research (in the field of CV in general including image classification). Socher et al. [SyLNM11] has attempted to combine color and texture features from over-segmented regions by Recursive Neural Networks. This work has been extended by another work of Socher et al. [SHB⁺12], which combined it with CNNs. With regards to segmentation, CNN-based approaches are the most popular as end-to-end supervised segmentation [GBC, FCNL13].

Farabet et al. [FCNL13] introduced multi-scale CNNs to learn scale-invariant features, but had problems with global contextual coherence and spatial consistency. These problems were addressed by combining CNNs with several post-processing algorithms, i.e., super-pixels, CRF, and segmentation trees.

The first post-processing strategy, *super-pixels*, which over-segments the relevant regions of an input image. From the class prediction produced by CNN, the average class distribution is computed within the super-pixel to aggregate the predictions of each super-pixel region (see Figure 2.6).

Since super-pixel-based post-processing takes only local dependencies within each super-pixel into account, *CRF*-based post-processing is introduced to involve a global understanding of the scene. First, a graph is constructed among pixels of an image, and then an optimal segmentation based on an energy function is found [SWRC06]. The energy function is composed of a unary and a pairwise term. The unary term is the output of super-pixel post-processing. The pairwise term $\Psi(l_i, l_j)$ is computed as follows:

$$\Psi(l_i, l_j) = \exp(-\beta \|\nabla I\|_i) 1(l_i \neq l_j), \quad (2.12)$$

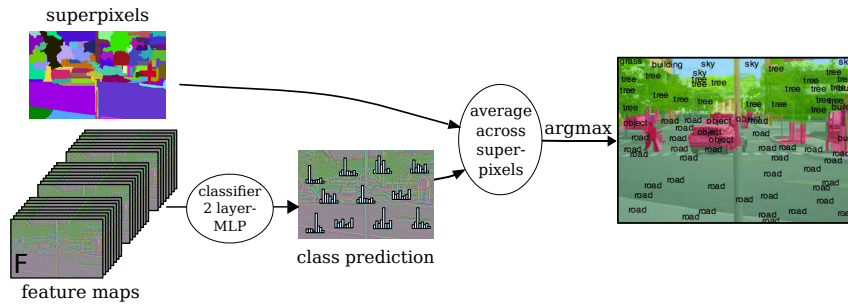


Figure 2.6: The first post-processing strategy from CNN using super-pixels. The class prediction of CNN is averaged within the super-pixel regions. The figure is reproduced from [FCNL13].

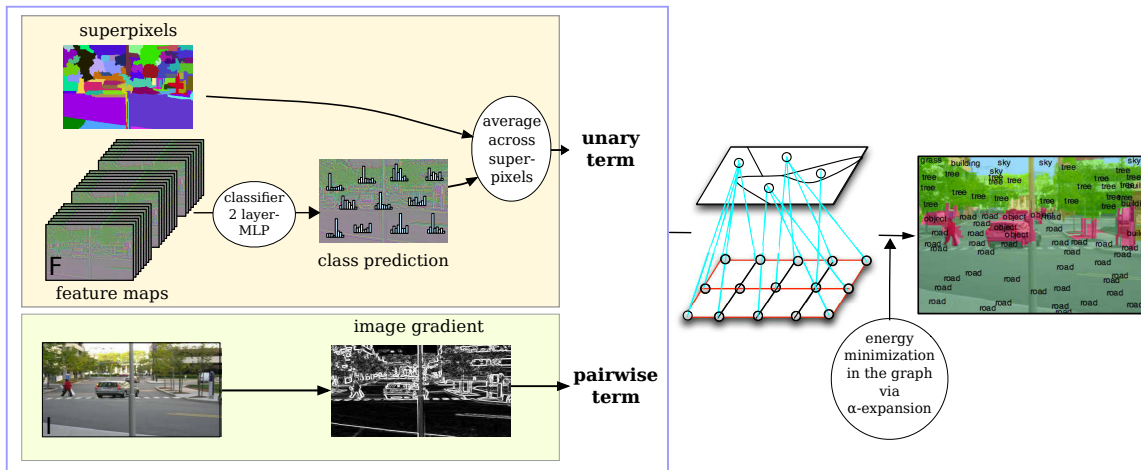


Figure 2.7: The second post-processing strategy of CNN using CRF. The graph is constructed within pixels. The energy function in CRF consists of unary term and pairwise term. The output of super-pixel post-processing explained above is used as unary term, and the pairwise term are defined using the $L2$ norm of the gradient of the image. CRF minimizes the energy function and finds the optimal graph. The figure is reproduced from [FCNL13].

where β is a constant, $\|\nabla I\|_i$ is the $L2$ norm of the gradient of the image I at the pixel i , $1(\cdot)$ is the indicator function which is 1 if the input is true, and 0 otherwise. The energy function is minimized using alpha-expansions [BK04]. An illustration of the procedure is shown in Figure 2.7.

These two methods explained above are based on arbitrary segmentation of the image. The third technique, *segmentation tree*, automatically analyzes the best level of compositions of an image using a segmentation tree. The components of the tree C_k are encoded by a component-wise max-pooling feature vector of a spatial grid (see Figure 2.8). A classifier (2 layer-MLP) is trained to estimate the histogram of

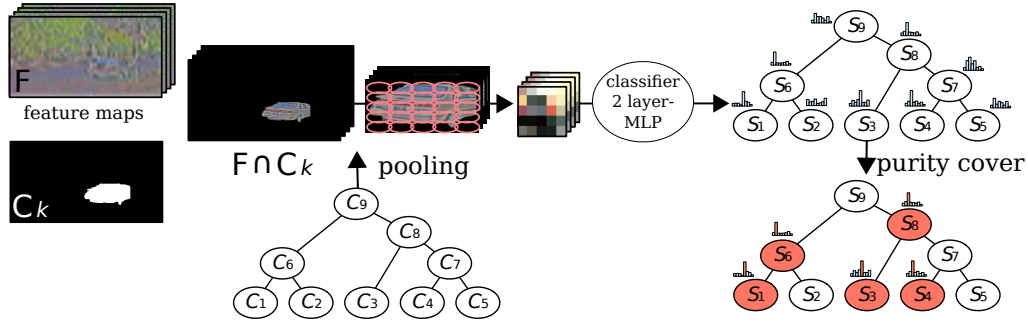


Figure 2.8: The third post-processing strategy of CNN using the segmentation tree. C_k is the k th component, and S_k is the cost associated with the predicted class distribution. At the end, each C_k , chosen based on the optimal purity cover is labeled using the predicted class distribution. The figure is reproduced from [FCNL13].

all categories from the component, which are then selected according to the purity cost. The purity cost is computed based on the entropy of the class distribution that find a consistent segmentation; pixels in one segment belong to only one category. Figure 2.8 illustrates the procedure of the segmentation tree approach. For more details of these three post-processing techniques, see the original work by Farabet et al. [FCNL13].

Later, Kekeç et al. [KEF⁺14] improved CNNs by combining two CNN models, learning context information and visual features from separate networks. Both mentioned approaches improved accuracy through carefully designed pre-processing steps to help the learning, i.e., class frequency balancing by selecting the same amount of random patches per class and by selecting a specific color space for the input data.

In order to improve modeling of long range dependencies, Pinheiro et al. [PC14] introduced RCNNs for scene labeling. They first revealed the use of large input patches to consider larger contexts. This, however, resulted in a reduction of the resolution of the final label image and a huge redundancy of overlapping regions making the learning inefficient. RCNNs train various sizes of the same input image (the instances) recurrently to learn increasingly large contexts for each pixel, whilst ensuring that the larger contexts are coherent with the smaller ones. Each instance is trained with the typical CNNs, the parameters (W, b) are shared between instances. The p th instance of the network F^p is defined as:

$$F^p = [f(F^{p-1}, I_{i,j,k}^p)], F^1 = [0, I_i, j, k], \quad (2.13)$$

where f is the output of the network, $I_{i,j,k}^p$ is the scaled version of $I_{i,j,k}^{p-1}$ at the pixel (i, j) of the image k . Here, the network of the current (p th) instance is trained with the feature maps of the previous $((p - 1)$ th) instance and the input image of the current instance.

Finally, the system maximizes the likelihood of all instances:

$$L(f) + L(f \circ f) + \dots + L(f \circ^p f), \quad (2.14)$$

where $L(f)$ is a likelihood (here, log-likelihood is used), \circ^p is the composition of the typical CNNs performed p times (see Figure 2.9).

To accurately localize the object segmentation, some of the recent research in the field combined CNN with CRF. Chen et al. [CPK⁺14] employed fully-connected CRF as post-processing on top of the probability map of CNNs. Their performance in the PASCAL VOC 2012 benchmark⁹, which is one of the most popular semantic image segmentation challenges, has achieved as the state-of-the-art method for image segmentation. Later, Zheng [ZJR⁺15] accomplished a complete end-to-end segmentation system by integrating CRF inside the framework.

2.4 Conclusion

Though CNNs were inspired by humans' visual mechanisms, there are several weaknesses to fulfill CV tasks. CNNs require specific kernel sizes, and these kernels only see local context. Therefore, the networks rely on certain scales of context and need further processing to combine with global contextual information if it is necessary for the specific task (as in image segmentation). LSTM takes the local and global contexts into account by nature in a single process, which overcome such issues. In the following chapters, the details of LSTM and how it is applied to image analysis tasks will be discussed.

⁹<http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?challengeid=11&compid=6>

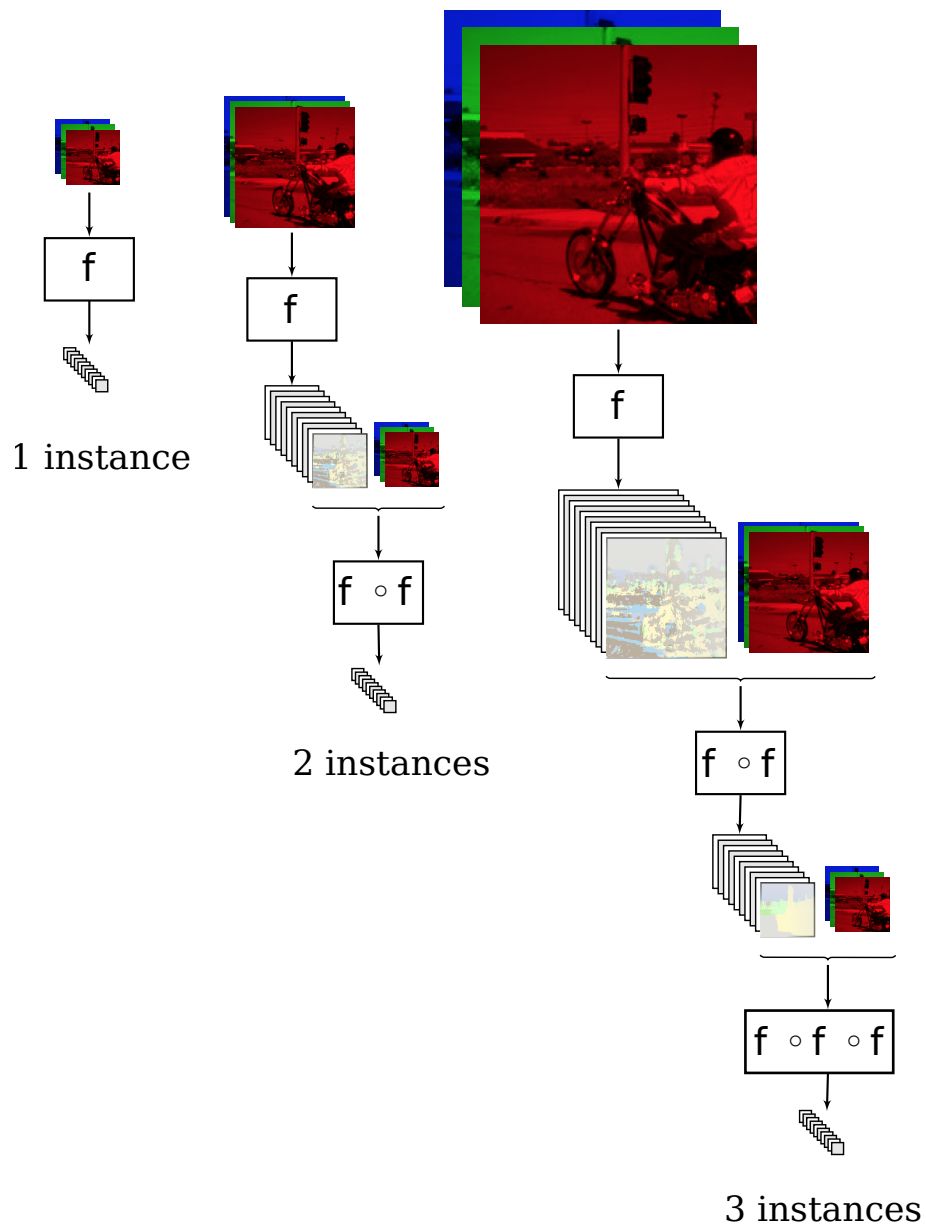


Figure 2.9: The composition of RCNNs. This figure illustrates the composition of one, two, and three instances. The figure is reproduced from [FCNL13].

Chapter 3

Multi-Dimensional LSTM (MD-LSTM) and Its Variant, PyraMiD-LSTM

This chapter presents two different LSTM models for 2D and 3D data, MD-LSTM and PyraMiD-LSTM. First, the background of LSTM and the traditional MD-LSTM for 2D data are described. Based on the traditional MD-LSTM model, a new parallelizable variant of MD-LSTM, PyraMiD-LSTM. Since the traditional MD-LSTM cannot be easily parallelized due to its architectural nature. This issue has been improved through the changes of the connection topology, which creates the independencies of a sequential flow. This model is efficient especially for 3D data.

Section 3.1 explains the background of MD-LSTM. First, the architecture of both RNN and BRNN, as well as their difference between them are discussed before presenting the standard LSTM. Section 3.2 explains MD-LSTM in details. In Section 3.3, a new LSTM model for 3D volumetric data is introduced¹.

¹This work presented in Section 3.3 in this chapter appeared in NIPS 2015 [SBLS15] and Marijn F. Stollenga was equally contributed to this work.

3.1 Background

This section describes the prior research related to MD-LSTM, which are RNN, BRNN, and traditional LSTM. These relevant studies with the detailed description provide how MD-LSTM is developed and improved over the past decade.

3.1.1 Notations

Here, the common notations used in this thesis are defined.

- an input sequence, $x = (x_1, x_2, \dots, x_{t-1}, x_t, x_{t+1}, \dots, x_T)$, where T is the length of the sequence
- a target sequence of the given input sequence,
 $y = (y_1, y_2, \dots, y_{t-1}, y_t, y_{t+1}, \dots, y_T)$, where T is the length of the sequence
- the predicted output of the given input sequence,
 $y^* = (y_1^*, y_2^*, \dots, y_{t-1}^*, y_t^*, y_{t+1}^*, \dots, y_T^*)$, where T is the length of the sequence
- (\cdot) is a matrix multiplication
- (\odot) is an element-wise multiplication

3.1.2 Recurrent Neural Networks

RNNs are neural networks, which are used for training with sequence data like speech or handwriting. Their self-hidden states (internal memory) store the temporal behavior of an input sequence and allow to predict the corresponding class labels based on the previous context of the sequence. The RNNs compute the current hidden state (h_t) by:

$$h_t = \phi(W \cdot x_t + H \cdot h_{t-1} + b), \quad (3.1)$$

where h_{t-1} is the recurrent hidden state of $t - 1$, which is a point of time lying in the past. W and H are weight matrices and b is a bias vector. ϕ is a non-linear activation function, usually logistic sigmoid (sigm) or hyperbolic tangent (tanh):

$$\text{sigm}(x) = \frac{1}{(1 + e^{-x})} \quad (3.2)$$

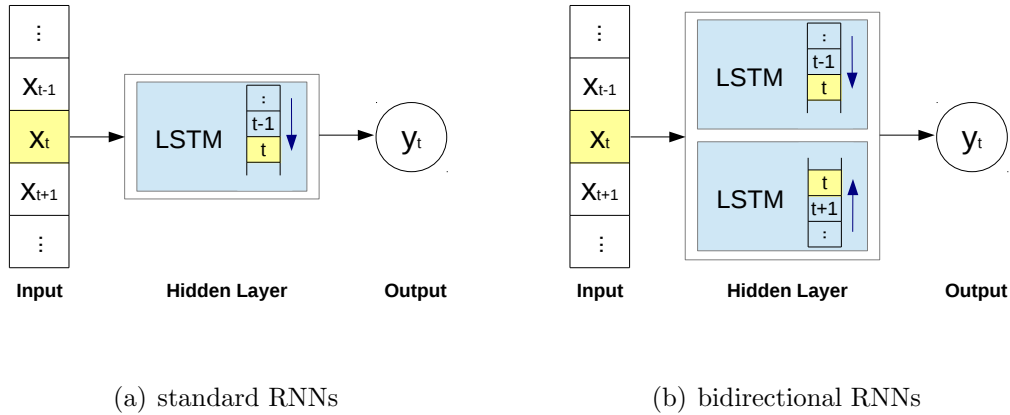


Figure 3.1: The architectural difference between RNNs and BRNNs. x_t is the input in current time, y_t is the corresponding output. The arrow shows the information flow of a LSTM module

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \quad (3.3)$$

An extension to BRNN [SP97] adds another self-hidden state for taking the other direction into account — the future time $t + 1$. Figure 3.1 shows the architectural difference between RNNs and BRNNs. The standard RNNs have one LSTM memory block to carry the past context of the sequence. However, the bidirectional RNNs pass through two memory blocks, which take both past and future contextual information into account. Therefore, BRNNs are effective when the length of the sequence is known.

However, the approach is still limited due to the vanishing gradient problem [BSF94, HBFS01]. The main problem is that the networks cannot capture long sequence dependencies; the gradient information either decays or blows up exponentially in case the input has more than 5-10 time lags [HS97b, Ger01].

3.1.3 Long Short-Term Memory (LSTM)

As discussed in the previous section, RNNs fail to learn long-term sequences. To avoid the problem, LSTM has been introduced by Hochreiter and Schmidhuber [HS97a]. Unlike RNNs, a memory block in LSTM has a self-hidden unit (memory cell) with a recurrent connection, and two gating units (input and output gates) which control

the access of information to the memory cell according to the previous context. Later, Gers et al. [GSC00] modified this initial architecture by adding a forget gate; it learns the behavior of the memory self-reset (forgetting). LSTM networks are successfully applied for sequence labeling such as off-line handwriting recognition [GLF⁺09] and speech recognition [SSB14].

This thesis follows the most commonly used architecture described in [Gra12]. A LSTM memory block includes three gates: an input gate (i), a forget gate (f), and an output gate (o), which overwrite, keep, or retrieve the memory from the memory cell (c) at the time t .

First, input gate (i_t) and forget gates (f_t) are computed by:

$$i_t = \text{sigm}(W_i \cdot x_t + H_i \cdot h_{t-1} + C_i \cdot c_{t-1} + b_i), \quad (3.4)$$

$$f_t = \text{sigm}(W_f \cdot x_t + H_f \cdot h_{t-1} + C_f \cdot c_{t-1} + b_f), \quad (3.5)$$

Afterwards, the current memory cell (c_t) is updated by an amount of the previous contents (c_{t-1}) for forgetting and the new memory (\tilde{c}_t) for including.

$$\tilde{c}_t = \text{tanh}(W_{\tilde{c}_t} \cdot x_t + H_{\tilde{c}_t} \cdot h_{t-1} + b_{\tilde{c}_t}), \quad (3.6)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \quad (3.7)$$

At the end, the final activation at the current position (h_t) is calculated with the output gate (o_t), which regulates the amount of information to output.

$$o_t = \text{sigm}(W_o \cdot x_t + H_o \cdot h_{t-1} + C_o \cdot c_t + b_o) \quad (3.8)$$

$$h_t = o_t \odot \text{tanh}(c_t), \quad (3.9)$$

$x, i, f, \tilde{c}, c, o, h \in \mathbb{R}^T$, where T is the length of the input. x_t is the input activation at the current time (t), and h_{t-1} is the output activations from the past time ($t-1$). W, H , and C are weight matrices for input to gates, recurrent connections, and cell to gates. The LSTM memory block is illustrated in Figure 3.2.

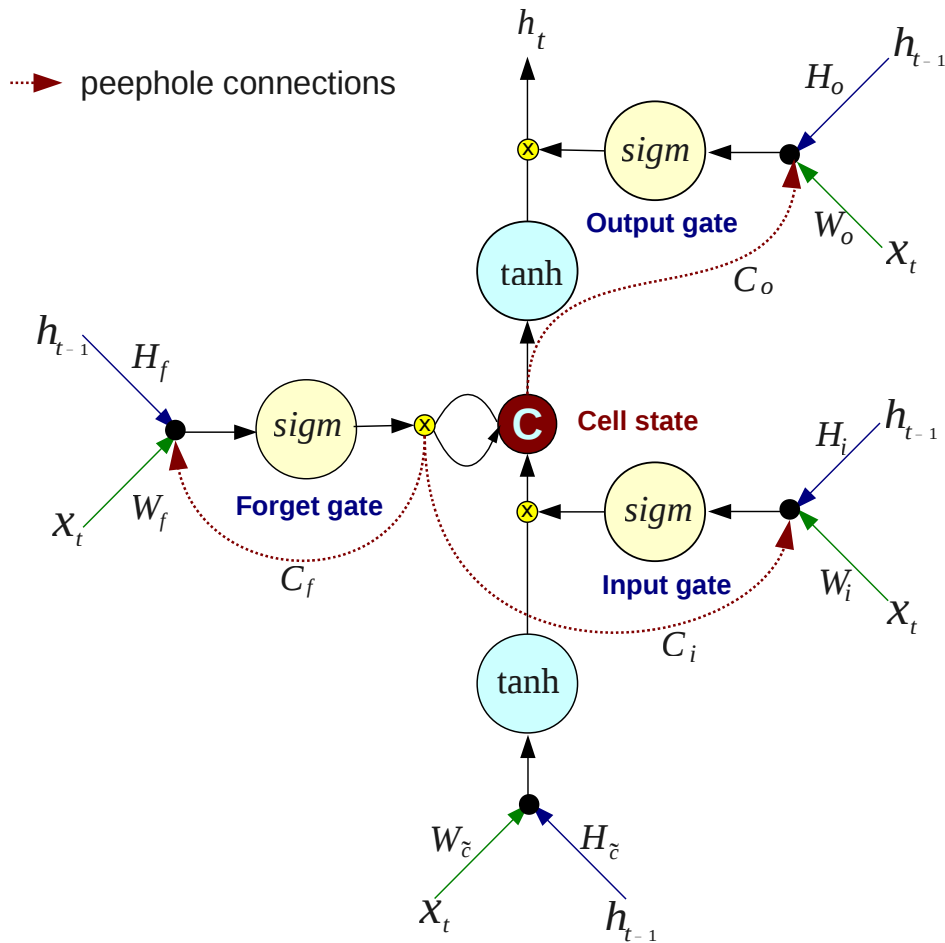


Figure 3.2: LSTM memory block.

3.2 Traditional Multi-Dimensional LSTM (MD-LSTM)

The LSTM model explained above is for one dimensional sequences; the model has one memory block with one self-recurrent connection. The main difference of MD-LSTM in the operations compared to the One Dimensional LSTM (1D-LSTM) is that multiple connections exist for each axis. These connections carry the neighboring contextual information. For instance on 2D images, there exist two self-recurrent connections which connect to the cell (x and y axes). Additionally, each LSTM module is computed independently to collect the surrounding contextual information in all directions: 2^{dim} LSTM modules (for 2D images $2^2 = 4$: top-left, top-right,

bottom-left, and bottom-right). In Figure 3.4–a, the connections and the context information flow of 2D-LSTM are illustrated.

The main idea of MD-LSTM has first been introduced by Graves et al. [GFS07a] and has been applied in many applications such as handwriting recognition [GS08] and binarization [GFS07a]. Recently, different ways of dealing with multi-dimensional data in the networks has been addressed [VKC⁺15, KDG15]. In this thesis, the standard 2D-LSTM architecture with peepholes described in [GS08] are mainly used.

Similar to 1D-LSTM, three gates (input (i), forget (f), output (o) gates) as well as the cell state (c) are computed over all pixels, recursively. Here, LSTM operations for one direction are summarized:

$$\begin{aligned}
 i_t &= \text{sigm}(W_i \cdot x_t + \sum_{p \in \mathcal{P}} (H_i^p \cdot h_{t-1}^p + C_i^p \cdot c_{t-1}^p) + b_i), & \text{(Input gate)} \\
 f_t^{p'} &= \text{sigm}(W_f \cdot x_t + \sum_{p \in \mathcal{P}} (H_f^p \cdot h_{t-1}^p) + C_f^{p'} \cdot c_{t-1}^{p'} + b_f^{p'}), & \text{(Forget gate for the axis } p') \\
 \tilde{c}_t &= \text{tanh}(W_{\tilde{c}} \cdot x_t + \sum_{p \in \mathcal{P}} (H_{\tilde{c}}^p \cdot h_{t-1}^p) + b_{\tilde{c}}), \\
 c_t &= \sum_{p \in \mathcal{P}} (f_t^p \odot c_{t-1}^p) + i_t \odot \tilde{c}_t, & \text{(Cell state)} \\
 o_t &= \text{sigm}(W_o \cdot x_t + \sum_{p \in \mathcal{P}} (H_o^p \cdot h_{t-1}^p) + C_o \cdot c_t + b_o), & \text{(Output gate)} \\
 h_t &= o_t \odot \text{tanh}(c_t), & \text{(Net-output)}
 \end{aligned}$$

where \mathcal{P} indicates the connections along with the axes (x and y for 2D). The 2D-LSTM memory block is illustrated in Figure 3.3.

As mentioned earlier, the computation above is for one direction (one LSTM memory block) of the context of the input. In other words, the final output h_t actually indicates h_t^d : $d \in \mathcal{D}$, where \mathcal{D} indicates the directions over the axes.

$$a_t = \sum_{d \in \mathcal{D}} h_t^d, \quad (3.10)$$

$$z_t = W_y \cdot a_t + b_y, \quad (3.11)$$

At the end, z_t is sent to the next layer.

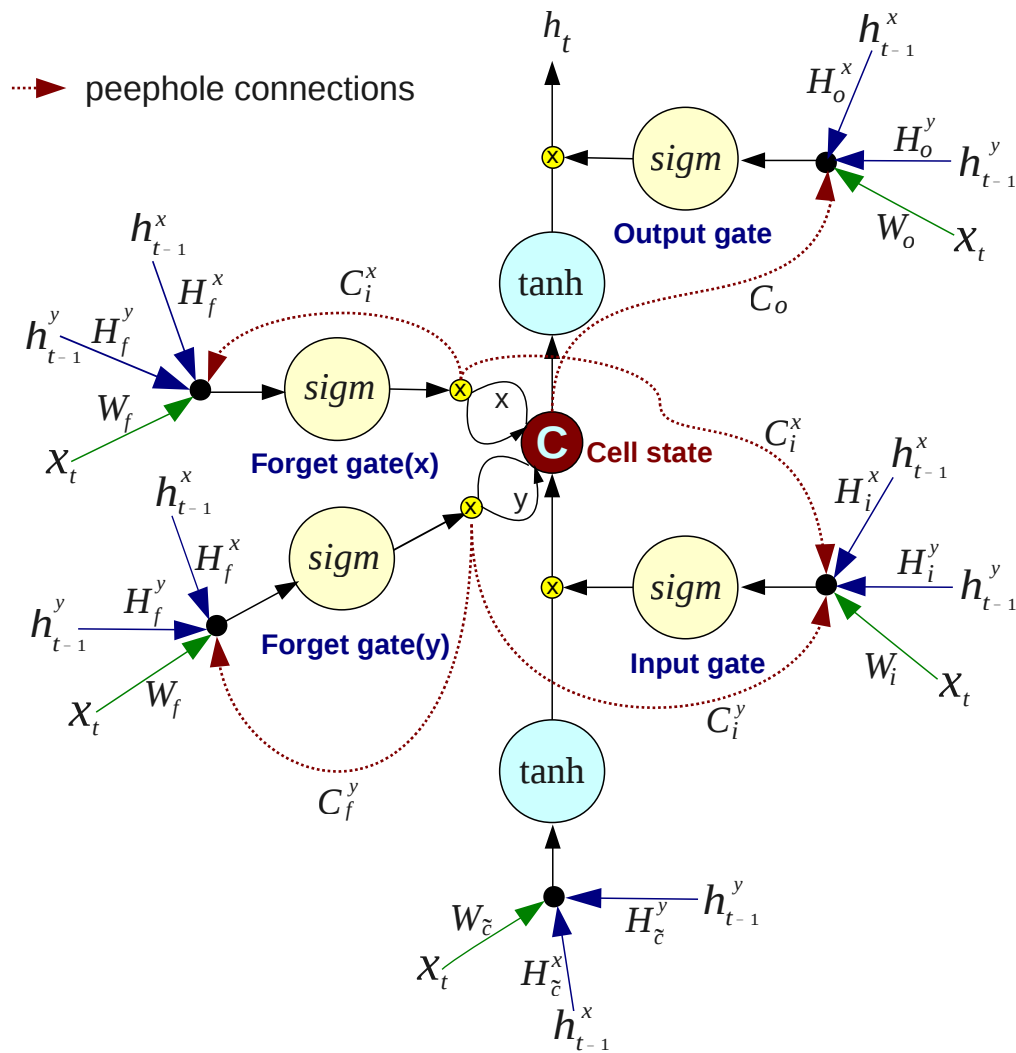


Figure 3.3: 2D-LSTM memory block.

3.3 Proposed PyraMiD-LSTM: Parallel MD-LSTM

Theoretically, the MD-LSTM model is capable of expanding the model to higher dimensions. However, in practice difficulties arise from the fact that an exponential amount of computations are needed; 2D-LSTM requires four LSTM modules, and 3D-LSTM needs 8 modules to cover all directions. Moreover, MD-LSTM cannot be easily parallelized, due to the sequential nature of RNNs. Therefore, a new and easily parallelizable LSTM model for more than two dimensional data is proposed;

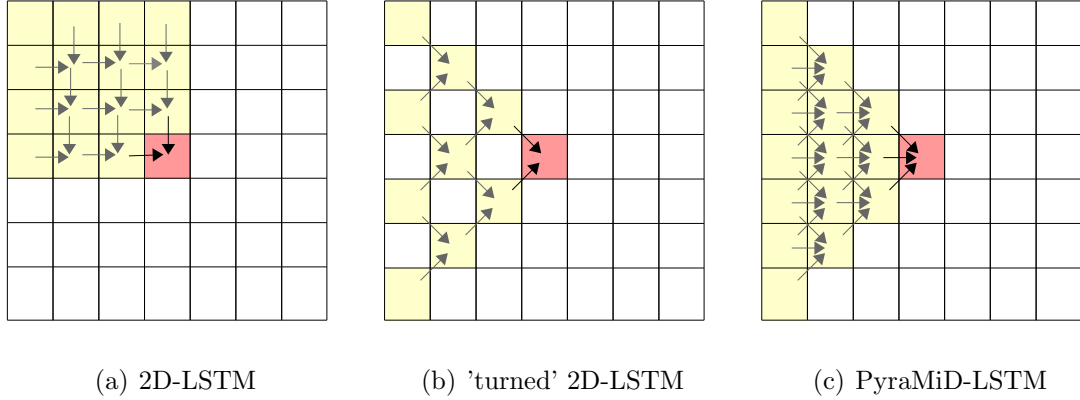


Figure 3.4: Recurrent connections and their context information flow of 2D and PyraMiD-LSTM. The red one is the current pixel. The arrow indicates the recurrent connections and corresponding context information flow. The current pixel implicitly carries the yellow pixels. **(a) 2D-LSTM:** The model evaluates the context of each pixel recursively from neighboring pixels along the axes. After turning the order by 45° like (b), dependencies of the current pixel become a plane (a column vector in the 2D case). **(c) PyraMiD-LSTM:** The gaps are filled by adding extra connections to process more than two elements of the context.

3D volumetric data is mainly considered here.

As mentioned earlier, MD-LSTM, aligns LSTM-units in a grid and connects them over the axis. Multiple grids are needed to process information from all directions. However, a small change in connections can greatly facilitate parallelization. If the connections are rotated by 45° , all input to all units come from either left, right, up, or down (left in case of Figure 3.4–b), and all elements of a row in the grid row can be computed independently. However, this introduces context gaps as in Figure 3.4–b. By adding an extra input, these gaps are filled as in Figure 3.4–c. Expanding this approach into three dimensions results in a Pyramidal Connection Topology. In other words, the context of a pixel is formed by a pyramid in each direction.

One of the striking differences between PyraMiD-LSTM and MD-LSTM is the shape of the scanned contexts. Each LSTM memory block of MD-LSTM scans rectangle-like contexts in 2D or cuboids in 3D. On the other hand, PyraMiD-LSTM scans triangles in 2D and pyramids in 3D (see Figure 3.5). MD-LSTM needs 8 LSTM memory blocks to scan a volume, while PyraMiD-LSTM needs only 6, since it takes 8 cubes or 6 pyramids to fill a volume. Given a dimension d , the number of LSTM memory blocks grows as 2^d for an MD-LSTM (*exponentially*) and $2 \times d$ for a PyraMiD-LSTM (*linearly*).

A similar connection strategy has been previously addressed [WDB⁺08]; the approach speeds up non-Euclidian distance computations on surfaces. There are however important differences:

- We can exploit efficient GPU-based CUDA convolution operations, which are different from the operations performed in CNNs.
- As a result of these operations in LSTMs, input filters that are bigger than the necessary 3×3 filters arise naturally, creating overlapping contexts. Such redundancy turns out to be beneficial and is used in our experiments.
- Several layers of complex LSTM processing with multi-channeled outputs and several state-variables in LSTM for each pixel are applied — instead of having a single value per pixel as in distance computations.
- Our application is focused on 3D volumetric data.

Here, the details of PyraMiD-LSTM with the computations for 3D volumes are described. The network consists of six LSTM memory blocks with RNN-tailored convolutions called Convolutional LSTM (C-LSTM), one for each direction, to create the full context of each pixel. Note that each of these C-LSTM is entire LSTM computations, processing the entire volume in one direction. The directions \mathcal{D} are formally defined over the three axes (x, y, z) : $\mathcal{D} = \{(\cdot, \cdot, 1), (\cdot, \cdot, -1), (\cdot, 1, \cdot), (\cdot, -1, \cdot), (1, \cdot, \cdot), (-1, \cdot, \cdot)\}$.

Each C-LSTM performs all computations in a plane moving into the defined direction. The input is $x \in \mathbb{R}^{W \times H \times D \times C}$, where W is the width, H the height, D the depth,

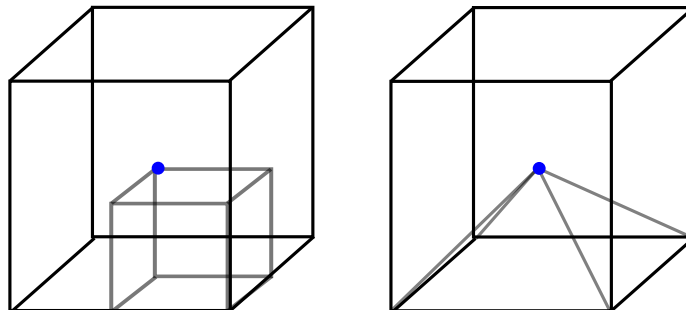


Figure 3.5: On the **left** we see the context scanned so far by one of the 8 LSTMs of a 3D-LSTM: a cube. In general, given d dimensions, 2^d LSTMs are needed. On the **right** we see the context scanned so far by one of the 6 LSTMs of a 3D-PyraMiD-LSTM: a pyramid. In general, $2 \times d$ LSTMs are needed.

and C the number of channels of the input, or hidden units in the case of second- or higher layers. Similarly, we define the volumes $f^d, i^d, o^d, \tilde{c}^d, c^d, h^d, h \in \mathbb{R}^{W \times H \times D \times O}$, where $d \in \mathcal{D}$ is a direction and O is the number of hidden units per pixel. Since each direction needs a separate volume, we denote volumes with $(\cdot)^d$.

The time index t selects a slice in direction d . For instance, for direction $d = (\cdot, \cdot, 1)$, v_t^d refers to the plane x, y, z, c for $x = 1..X, y = 1..Y, c = 1..C$, and $z = t$. For a negative direction $d = (\cdot, \cdot, -1)$, the plane is the same but moves into the opposite direction: $z = Z - t$. A special case is the first plane in each direction, which does not have a previous plane, hence we omit the corresponding computation.

C-LSTM equations:

$$\begin{aligned}
 i_t &= \text{sigm}(x_t * W_i + h_{t-1} * H_i + b_i), && \text{(Input gate)} \\
 f_t &= \text{sigm}(x_t * W_f + h_{t-1} * H_f + b_f), && \text{(Forget gate)} \\
 \tilde{c}_t &= \text{tanh}(x_t * W_{x\tilde{c}} + h_{t-1} * H_{\tilde{c}} + b_{\tilde{c}}), \\
 c_t &= \tilde{c}_t \odot i_t + c_{t-1} \odot f_t, && \text{(Cell state)} \\
 o_t &= \text{sigm}(x_t * W_o + h_{t-1} * H_o + b_o), && \text{(Output gate)} \\
 h_t &= o_t \odot \text{tanh}(c_t), && \text{(Net-output)}
 \end{aligned}$$

where $(*)$ is a convolution and h is the output of the layer. Note that in 3D volumes, convolutions are performed in 2D; in general a n-D volume requires n-1-D convolutions. All convolutions have stride 1, and their filter sizes should at least be 3×3 in each dimension to create the entire context. All biases are the same for all LSTM units (i.e., no positional biases are used). Further processing is the same as Equation 3.10 and Equation 3.11.

3.4 Conclusion

Here, the traditional MD-LSTM and its improved version, PyraMiD-LSTM are presented. In the next section, how these two LSTM models are employed in the network, the details of each layer of the networks, and their architectural choices for image analysis will be described. Also, the details of network settings and the generalization techniques for the network training will be explained.

Chapter 4

Network Architectures for Image Analysis

This chapter describes each layer of the LSTM network and provides a number of network architecture options for MD-LSTM-based image analysis and possible network settings for efficient training. All architectural options of the network and settings described here are subsequently analyzed.

Section 4.1 explains the details of possible layers in the LSTM network. In Section 4.2, the design choices with these layers for image analysis are described. Finally, the possible network settings and generalization techniques of the networks are described in Section 4.3.

4.1 Network Layers

In this section, all possible architectures combined with various layers are presented with a focus on CV tasks. The main architecture of the network in this thesis can be divided into three layers: an input layer, a hidden (LSTM) layer, and an output layer. Additional layers such as a fully connected layer and a collapse layer can be included in the hidden layer and the output layer, respectively. Two different target coding schemes, i.e. a standard and a probabilistic target coding scheme, are introduced and applied depending on the task and the input.

4.1.1 Input Layer

As in other DL approaches, the LSTM networks directly receive raw pixel values as input; the image can be a grayscale or color image, but here only an RGB color image is used as input. The input ($3 \times n \times n$) can be one pixel ($n = 1$) or a window ($n > 1$). When the input is a single pixel, it will result in having a very long sequence of entire pixels to learn in each iteration. In the worst case, LSTM is required to carry the contextual information of whole pixels to learn dependencies. Though the issue of long-term dependencies is eased in LSTM (compared to RNNs), Hochreiter found that LSTM can learn to bridge between 1000 discrete steps at most [HBFS01].

Since, adjacent pixels in an image, especially with a high-resolution, tend to have similar values, the dependencies within the pixels can be redundant. Based on this assumption, the window-input is utilized depending on the task and the resolution of the image. Each window can be viewed as a super-pixel. This super-pixel-input combines the local correlation of the pixels but maintains global coherence of the image. If images are present in high resolution and quality, the window-input integrates the local context and reduces the total discrete input steps. It helps in both localizing the area and keeping the longer range dependencies without losing global context. Moreover, the window-input speeds-up the inference process.

For classification, since the precise positioning is not necessary as an output, the window-input is effective regardless of the resolution of the image. However, the window-input at the end affects the size of the final output of the segmentation¹. The final size of the output matters for the image segmentation task. This issue can easily be solved by up-scaling with some interpolation method, e.g., cubic interpolation. Thus, the issue does not heavily influence the final results. Nevertheless, if the size of the total input is small enough, pixel-level input is still better as it provides more precise output position than the output of an interpolation. In this thesis, window-input is applied, if the resolution of the input image is higher than 100×100 .

Another issue concerning the window-input in segmentation is that one window-input with many pixels is compared to one target label in a typical error computation, meaning that a single label from these pixels should be selected. In Section 4.1.3, a probabilistic target coding scheme is introduced to resolve this problem.

To obtain windows from the input, an image is split into a grid, which can be thought

¹Each window is assigned to a label instead of a pixel

of as non-overlapping windows. Initially, both overlapping and non-overlapping window-input was tested, and it was found that the latter performed just as well, but faster.

4.1.2 Hidden Layer

The main parts of hidden layers are LSTM memory blocks. In addition, fully connected layers can optionally be added for deeper networks.

LSTM Layer

The details of an LSTM layer are described in the last sections (mainly in Section 3.2 and 3.3). Each LSTM memory block is computed for all directions before the output activations are combined into one final output to send it to the next layer. This layer can be hierarchically stacked for a deeper model.

Fully Connected Layer

This layer has full connections with a nonlinear activation function. The output of the connections is combined and squashed by the hyperbolic tangent (\tanh). As indicated by Graves [Gra12], this layer helps to control the number of weights for the next layer. It also controls (mainly increases) the amount of features from the results of LSTM layers and provides a smoother information flow to the next layer. This layer is especially helpful in case of complex input and deeper networks. In Figure 6.5 of Section 6.2.1, some examples of feature maps from this layer are visualized.

4.1.3 Output Layers

Collapse Layer

This layer is mainly used for *Image-level Classification*. The main idea is initially introduced by Graves [Gra08], but has not been used. From the hidden layer, an input (a pixel or window) has its own prediction. The required output of image classification is a single label of an input image. Therefore, all of these predictions need to be contributed to a single class label at the end. This layer is mainly for integrating all predictions of an input pixel or a window into one final output. This integration occurs between the hidden layer and the classification layer, i.e., softmax layer. All activations over the whole input are added and sent to the final layer. The

operation is the sum over all the pixels (input):

$$n = \sum_{t=1}^T p_t, \quad (4.1)$$

where T is the length of input, p is the output from the hidden layer, and n is the final activation vector.

Softmax Layer

Finally, the activations from the last layer, contained in M , are fed to a softmax layer, which will output the class probabilities for input w_i : $Pr(c|w_i) = Softmax(M)$.

The output of the last hidden layer is normalized with the *softmax* function:

$$Pr(c|w_i) = \frac{e^{a_c(w_i)}}{\sum_{l \in \{1, \dots, L\}} e^{a_l(w_i)}},$$

where $a(w_i)$ is the final activation of the input w_i , c is the target class of the input, and l is the one of the predefined target sets. Here, w_i is either an input image (for image-level classification: image classification) or a pixel/window (for pixel-level classification: image segmentation) depending on the required output of the task.

Our goal is to find the maximum likelihood of all training samples. Various objective functions (loss/error function) can be used such as the cross-entropy error function or the squared error function. As the cross-entropy error function is most commonly used throughout this thesis, it will be further explained in the following:

$$E = - \sum_{c=1}^C z_c \ln Pr(c|w_i), \quad (4.2)$$

where $z_c \in \{0, 1\}$ is an integer value from the true probability vector corresponding to the target class c . Note that, the true probability vector is constructed based on its target class; 1 is assigned for the target class and 0 for others. $Pr(c|w_i)$ is the predicted probability of the class c .

Standard target coding: To compare the multi-class probability ($Pr(c|w_i)$) with the target c for errors, a *1-of-K coding scheme* is typically used, which It encodes the desired output. Note that, 1-of-K coding scheme is a binary vector with all

elements set to zero except for the one which corresponds to the correct class. In the Equation (4.2), z_c is an element of the encoding vector according to c .

Probabilistic target coding: In general, the number of targets corresponds to the length of the input. In image segmentation however, when the input is not one pixel, the number of target labels is greater than one. In this case, the target is not only a single class, but can contain multiple classes. Let us assume that the input is a window, sized $n \times n$, $n > 1$. In this case, errors are quantified within this window using a *probabilistic target coding scheme*. The error function (Equation (4.2)) is modified using the probability of an occurrence of the class c .

$$E = - \sum_{c=1}^C \frac{f_c}{n \times n} \ln Pr(c|w_i), \quad (4.3)$$

where f_c is the frequency of occurrence for class c in w_i , and $n \times n$ is the size of w_i .

4.2 Network Design for Image Analysis

This section explains the network design choice for various CV problems that are addressed in this thesis. The major concern of the network architecture design is to determine the kind of task the network aims for. Here, the architectural options are suggested depending on the type of input and output, as well as the required tasks. Thus, the network architectures are constructed based on three factors: the dimension of input, depth of a network, and the type of output. As mentioned earlier, the typical networks are with three layers: an input layer, a hidden layer with LSTM, and an output layer.

4.2.1 The Dimension of Input

This work focuses on processing multi-dimensional input (2D and 3D). When dealing with multi-dimensional data, the traditional approaches, e.g., MLP, pre-process the data into one dimension before sending it to the network. To deal with high-dimensional data without any extra processing, the LSTM layer takes a wide range of

contextual information in MD-LSTM. Here, two different LSTM connection strategies for the context flow of information are introduced: axis-wise connections (Figure 3.4-a) and column-wise connections (Figure 3.4-c). Depending on the dimension of input, one connection strategy is more efficient than the other.

2D input data: For 2D data, the traditional MD-LSTM memory block includes two recurrent connections as shown in Figure 3.4-a. These two recurrent connections access the pixels along the x and y axes. Thus, one memory block takes one direction of the image into account. For all directions, four LSTM memory blocks are necessary: left-top, left-down, right-top, and right-down. These access directions can be formally defined over the two axes (x, y) : $\mathcal{D}_{2D}^a = \{(-1, -1), (-1, 1), (1, -1), (1, 1)\}$. These memory blocks and their connections in the network are shown in Figure 4.2-a, b, c (in the LSTM Layer).

There is another possibility to process 2D data, which uses to have column-wise connections. In this case, at least three recurrent connections are required along with an axis (x or y , see Figure 3.4-c). It also requires four LSTM memory blocks in the 2D case: left, right, top, down. The access directions can be formally defined as follows $\mathcal{D}_{2D}^b = \{(\cdot, -1), (\cdot, 1), (1, \cdot), (-1, \cdot)\}$.

The latter strategy seems to require more computations than the former one; both strategies need to process four directions but the former one has two connections and the other has three connections. However, the LSTM computations with the column-wise connection can be efficiently computed with the convolution operation. This characteristic is especially advantageous when the dimension is higher, which will be discussed with 3D data in the following. In the experiments with 2D data, this thesis employs the MD-LSTM with axis-wise connections in the LSTM Layer.

3D input data: In traditional MD-LSTM for 3D data, three recurrent connections are needed along with the axes: (x, y, z) . This requires eight access directions to process the whole context. These access directions can be formally defined as: $\mathcal{D}_{3D}^a = \{(-1, -1, -1), (-1, -1, 1), (-1, 1, -1), (-1, 1, 1), (1, -1, -1), (1, -1, 1), (1, 1, -1), (1, 1, 1)\}$.

The connections with column-wise dependencies in 3D data become plain-wise connections. The number of connections is typically nine for one direction, and six LSTM

memory blocks are required to process all directions. As mentioned earlier, LSTM computations with this connection strategy are convertible to the convolution operations. The number of connections are decided by the filter size; the filter size 3×3 has nine connections. All directions with plain-wise connections can be defined as follows: $\mathcal{D}_{3D}^b = \{(\cdot, \cdot, -1), (\cdot, \cdot, 1), (\cdot, -1, \cdot), (\cdot, 1, \cdot), (-1, \cdot, \cdot), (1, \cdot, \cdot)\}$. This connection strategy is used in PyraMiD-LSTM. refer to Section 3.3 for more details.

In fact, MD-LSTM stands in need of tremendous computations to process all directions, especially for higher dimensions and a big data input. Therefore, the former connection strategy, which the parallel computation is not possible due to the dependencies of each pixel, is a critical limitation for 3D data. The latter connection strategy can easily solve the issue with parallel convolution computation by utilizing the GPU. Figure 4.2–d (in the LSTM Layer) illustrates the plain-wise connections and their context information flow on 3D volumetric data. Later in Chapter 7, we explain how the parallel volumetric LSTM network (PyraMiD-LSTM) is applied to 3D volumetric images.

4.2.2 Depth of the Network

Depth has an important influence on the performance of the network. This thesis investigates both a single-layer and a deep (multi-layer) network for image analysis tasks.

A single-layer network: This is a network with one hidden layer. Since building deep representation is an issue in CV and ML communities, a single-layer network does not seem attractive. Deep representation is known to provide different level of abstraction of images that is able to have automated feature learning. However, it needs a lot of computations (parameter weights) and is hard to train; the issue was discussed in Chapter 2. Instead, LSTM processes the range of contextual information recursively. By making use of the memory cell and gates, the LSTM memory block decides to remove or keep information of each pixel. In addition, the multiple independent memory blocks of MD-LSTM take various directions of an image into account. This structure is theoretically able to extract the abstract representation of an input image. The experiments in this thesis will show that a single-layer MD-LSTM network can be successfully applied to noisy texture and natural scene

images (in Section 5.1, Section 5.2, and Section 6.1).

A deep network: A single-layer LSTM network has achieved good experimental results in many challenging image analysis tasks in this thesis. However, to classify input data containing a high intra-class variation and noise, higher-level abstraction modeling is required for a more generalized system. Moreover, LSTM tends to struggle with a long-term dependencies meaning that the training becomes harder with increasing length of the data; the long-term dependency problem is addressed in Section 3.1.3. These issues are reduced with a deeper model. In this thesis, the deep model consists of three stacks of both the LSTM and the fully connected layer. Note that the fully connected layer is added between two LSTM layers, and the size of the fully connected layer is bigger than that of the LSTM layer. Hence, more combinations of low-level representations from the previous LSTM layer are produced. This helps to provide more diverse abstract representations to the deeper layer and to control the number of weights to the next layer. Figure 4.1 presents the internal representation of each layer of the three-layered networks after convergence, including all of the LSTM memory blocks as well as the fully connected layers.

4.2.3 The Type of Output

Depending on the shape or the type of the required output, the output layer needs to be designed accordingly. In this thesis, two domains of image analysis are addressed: image classification and segmentation. Image classification requires output with one desired label per image, but segmentation requires a set of desired labels per image which should be the same size as the input image. Here, the issue is solved with adding a layer before the final classification layer.

For Image classification: Since the LSTM layer results in an output activation vector (the class probabilities) for each pixel, the network needs to finalize the highest score of the class over all pixels for the task. Therefore, the entire output is integrated into a collapse layer after the hidden layer. The output of this layer is sent to the output layer, commonly a softmax layer. Finally, the main architecture for image classification includes an input layer, one or multiple hidden layers, a collapse layer, and an output layer. The architecture for the task is illustrated in Figure 4.2–a.

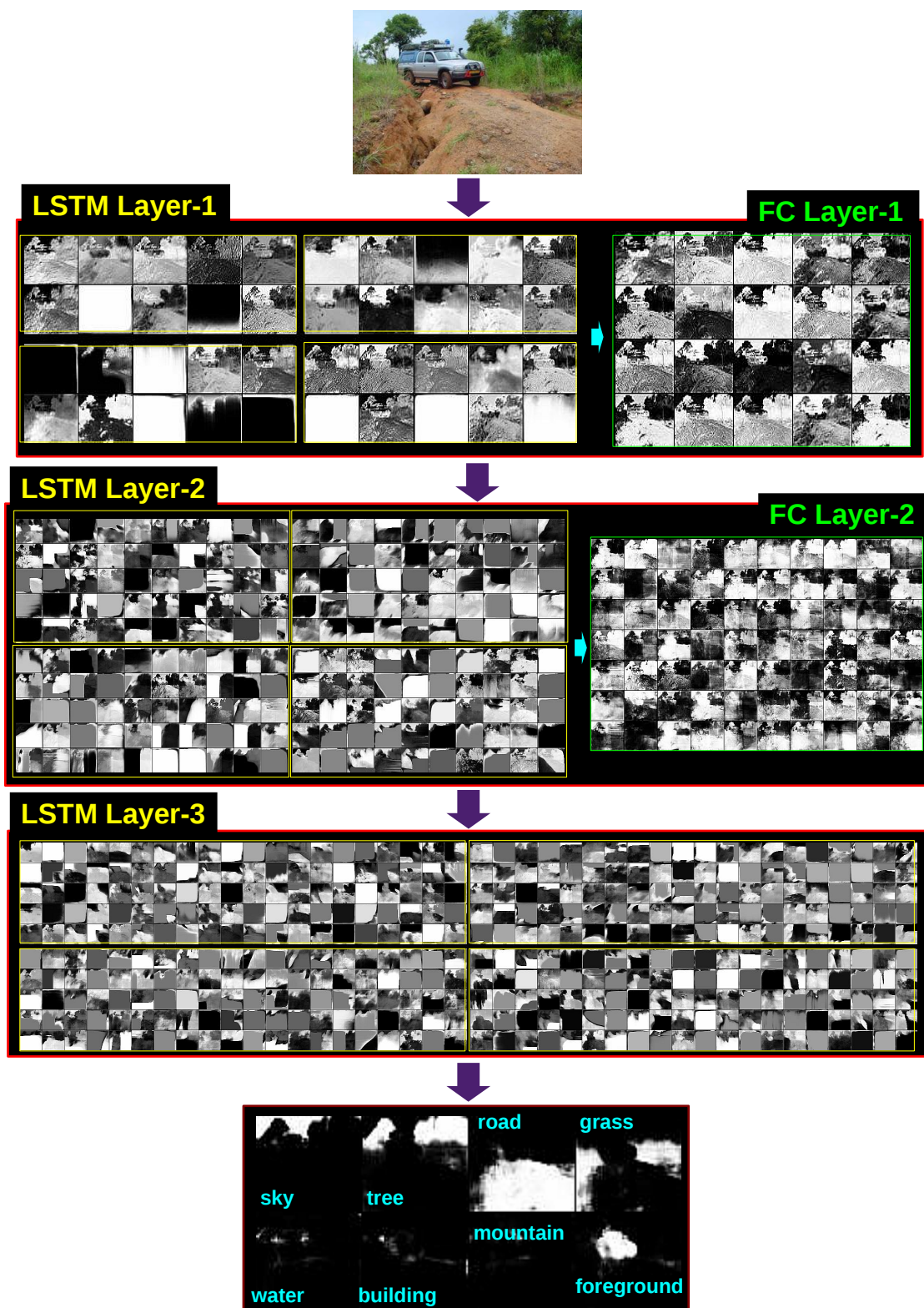
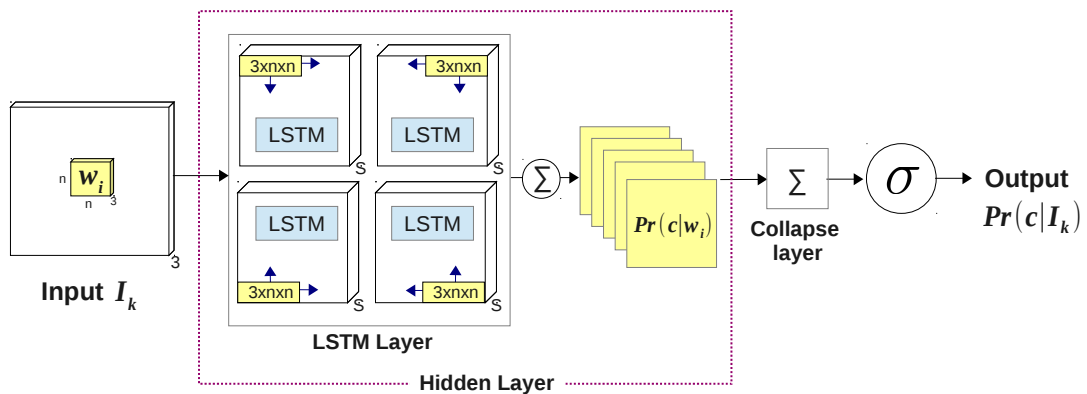
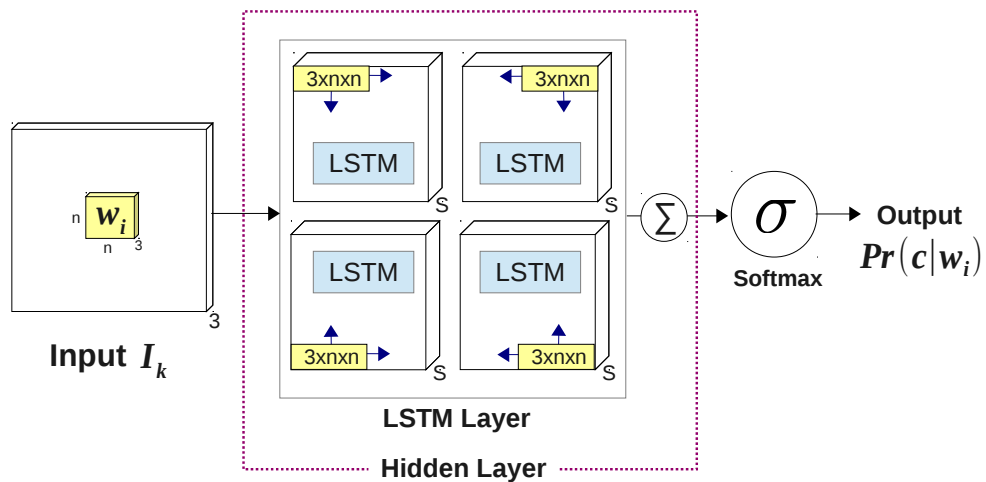


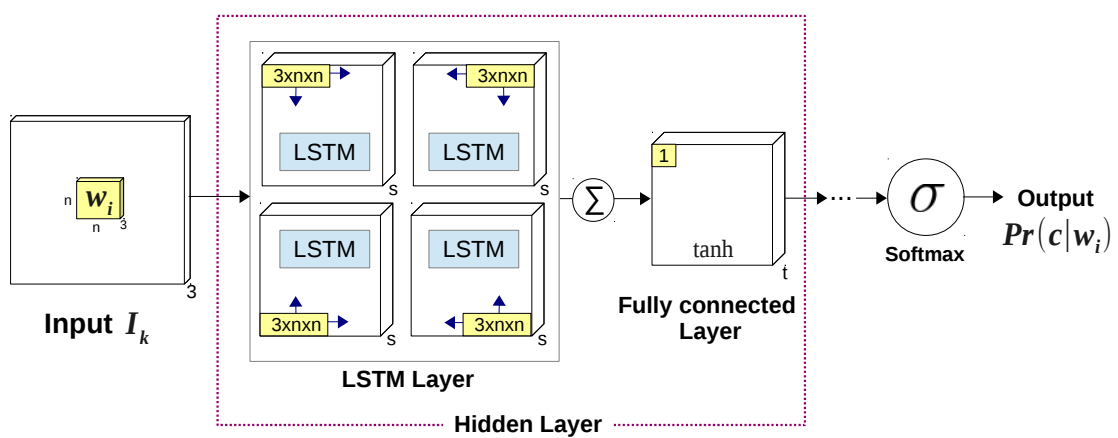
Figure 4.1: The internal representation of each layer of the three-layered networks. The networks are composed of the input layer, (the first) LSTM layer with a fully connected layer, (the second) LSTM layer with a fully connected layer (FC layer), (the last) LSTM layer, and the output layer with softmax. The lighter color represents higher activations.



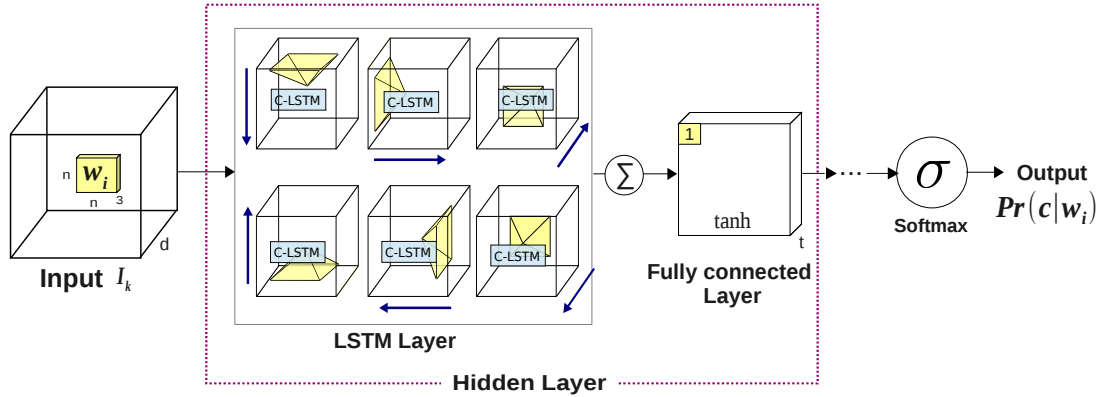
(a) 2D image classification



(b) 2D image segmentation (a single hidden layer)



(c) 2D image segmentation (multiple hidden layers)



(d) 3D image segmentation (multiple hidden layers)

Figure 4.2: Various architectures used in this thesis for different applications. s and t are the number of hidden units in each layer, which is not visualized in (d). d is the depth of volumetric data. The input consists of RGB images. Hence, the size of the input is always 3.

For Image segmentation: One of the advantages of LSTM for images is that the model delivers a per-pixel prediction without any extra processing. Therefore, the network does not need any extra layer. The main layers for image segmentation are an input layer, one or multiple hidden layers, and an output layer. The output of LSTM is directly sent to the output layer. Figure 4.2–b, c, d shows the architectures used in this thesis for image segmentation.

Figure 4.2 summarizes the different architectures used in this thesis for various CV tasks. Figure 4.2–a and b visualize a single-layer network for image classification and segmentation of 2D texture images, respectively. Figure 4.2–c depicts a deep network for image segmentation on 2D scene images. Finally, Figure 4.2–d represents for 3D volumetric segmentation with a deep network.

4.3 Network Settings and Generalization

There can be different settings and generalization techniques of the networks depending on their conditions and tasks. This section discusses the possible network settings and generalization techniques to optimize the final performance: input nor-

malization, weight initialization, peephole connections, regularization, optimization for training, and other network parameters.

4.3.1 Input Representation

Keeping an appropriate range of input values is essential for any machine learning tasks, especially for NNs [LBOM98]. There are several ways of changing a range of input values: scaling, normalization, standardization.

Scaling is to add or subtract input values by a pre-defined constant before multiplying or dividing with another constant. One example is to make a consistent unit of measurement from different types of data.

Normalization can be viewed as a way of scaling. This computation provides a range of input values in certain scales. One of the common normalization techniques is the Max-min normalization.

$$\bar{x}_t = \frac{x_t - x_{min}}{x_{max} - x_{min}}, \quad (4.4)$$

where x_{max} and x_{min} are the maximum and minimum values over the given input, and \bar{x}_t is the normalized input of x_t . This normalization technique keeps all input values in the range of $[0, 1]$, however it is also possible to keep the middle of the range at 0 and the interval of size 2 by using the following formula:

$$\bar{x}_t = \frac{x_t - ((x_{max} - x_{min})/2)}{(x_{max} - x_{min})/2}, \quad (4.5)$$

Here, the normalized input \bar{x}_t will be in the range of $[-1, 1]$.

Standardization (also called Z-score Normalization) centers the values around 0, i.e. keeping a mean of 0, with a standard deviation of 1 by calculating mean and standard

deviation (Z-score) over the training data.

$$\mu = \frac{1}{T} \sum_{i=1}^T x_i, \quad (4.6)$$

$$\sigma = \sqrt{\frac{1}{T} \sum_{i=1}^T (x_i - \mu)^2}, \quad (4.7)$$

$$\bar{x}_t = \frac{x_t - \mu}{\sigma}, \quad (4.8)$$

where μ and σ compute the mean and the standard deviation of the input data, respectively.

The purpose of this process is to provide compact but distinctive components of the input vectors between classes. It mainly performs uniform scaling to keep a certain range of variation in the input space. Therefore, the training becomes faster and reduces the chance of local optima. The choice of this process depends on the type of input, the activation functions (in NN) and the range of weight values; the weight initialization scheme is discussed in Section 4.3.2. A suitable range of input values with the corresponding weights and biases can have a huge effect on network performance. If not specified otherwise, this thesis assumes that the input to be normalized into the range $[0, 1]$. Standardization is applied in some experiments, especially for natural scene images, to maximize the variance of pixel values. However, standardization had no effect on performance and speed.

4.3.2 Weight Initialization

The initial weights can significantly affect the training process [LBOM98]. In general, weight values are randomly chosen from small values. In fact, the values should not be in a certain range to avoid very small gradients — small gradients slow down the training process. This thesis follows the same weight initialization as the one from Gers et al. and Graves [GSS02, Gra12]: Gaussian distribution with a mean value of 0 and a standard deviation of 0.1. For LSTM, all gate activations and biases are initialized to zero. As mentioned earlier in Section 3.1.3, it is hard to learn long-term dependencies even with LSTM. We assume that using a high-quality image as an input tends to raise a long-term dependency issue as the length of the input is longer

than the length of other data types, e.g., off-line handwriting. Therefore, some extra experiments are performed to see the effect of biasing forget gates with regards to the issue. Higher bias values are set at the beginning of the experiments, i.e., +1.0, +2.0, or +7.0. However, these initial bias setting did not affect performance.

4.3.3 Peephole Connections

The initial LSTM model proposed by Hochreiter et al. [HS97a] and some recent studies [LZCR15] do not have peephole connections which connect from a cell to a gate (i.e., $C_i \cdot c_{t-1}$, $C_f \cdot c_{t-1}$, and $C_o \cdot c_{t-1}$, see Section 3.2). Omitting them did not result in any performance penalties in experiments of this thesis, but resulted in less computations and therefore in a shorter processing time. The connections were kept for most of the experiments (classification and segmentation tasks) but discarded for the later work (volumetric segmentation).

4.3.4 Regularization

Several regularizers are introduced to address the problem of overfitting, such as Dropout [HSK⁺12] and DropConnect [WZZ⁺13]. The main idea of these approaches is to randomly remove a subset of unit activations (Dropout) or connections (DropConnect) in each layer. The networks with these regularizers are forced to learn very sparse representations, which results in a more robust training. They appear to be very effective at improving the quality of predictions. Alternatively, L1 and L2 regularization and weight decay can be used [Sla14]. In this thesis, Dropout was tested with the PyraMiD-LSTM model; limited to non-recurrent connections (50% dropout on fully connected layers and/or 20% on input layer), but it showed no effect on performance.

4.3.5 Optimization

There are several ways of optimizing a set of weight parameters so that the error based on the gradient information is minimized². There is a range of different minimiza-

²The commonly used error functions are Mean Square Error (MSE) or Cross-Entropy Error (Equation 4.2)

tion approaches investigated by the literature. Some of the examples are Stochastic Gradient Descent (SGD), momentum, RMS-prop [TH12], AdaGrad [DHS11], or combinations of them. This thesis has mainly been experimenting with on-line³ SGD with momentum and RMS-prop with momentum.

SGD is the most common way of updating weights in NNs. It computes the gradient of the parameters using only a single (on-line) or a few training examples (mini-batch).

The formula is as follows:

$$\theta = \theta - \lambda_{lr} \nabla_{\theta} E(y, y^*), \quad (4.9)$$

where y is the target, y^* is the predicted network output, E is the predefined error function. ∇ is the gradient, θ is the weight, and λ_{lr} is the learning rate.

This way, the update steps per iteration can be unstable, leading to local optima or poor convergence. This issue can be avoided by using momentum with SGD.

Momentum uses the history of the previous update V and combines it with the next update. This results in a more stable update compared to the standard SGD, as it can prevent some uncertain jumps.

The formula of the SGD with momentum is as follows:

$$V = \alpha V + \lambda_{lr} \nabla_{\theta} E(y, y^*), \quad (4.10)$$

$$\theta = \theta - V, \quad (4.11)$$

where α is the momentum coefficient.

RMS-prop with momentum is another extension to overcome the problem of the standard SGD. Using an average of squared gradients, it attempts to reduce a monotonically decreasing learning rate, meaning that even weights with small gradients get updated. This also helps to deal with vanishing gradients [Hoc91]. Let us define $a \stackrel{\rho}{\leftarrow} b$ to be $a_n = \rho a_n + (1 - \rho)b_n$, where $a, b \in \mathbb{R}^N$.

³The weights are updated per sample.

The following equations hold for every epoch:

$$\text{MSE} \stackrel{\rho_{\text{MSE}}}{\leftarrow} \nabla_{\theta}^2 E(y^*, y), \quad (4.12)$$

$$G = \frac{\nabla_{\theta} E(y^*, y)}{\sqrt{\text{MSE} + \epsilon}}, \quad (4.13)$$

$$M \stackrel{\rho_M}{\leftarrow} G, \quad (4.14)$$

$$\theta = \theta - \lambda_{\text{lr}} M, \quad (4.15)$$

where MSE represents a running average over the variance of the gradient, ∇^2 is the element-wise squared gradient, G the normalized gradient, ϵ is the smoothing constant to prevent division by zero, and M the smoothed gradient.

In this work, networks will mostly be trained with SGD with momentum, with the exception of 3D biomedical image segmentation which uses RMS-prop with momentum.

4.3.6 Network Parameters

The *Learning rate* is commonly set to a very small value. In general, choosing the appropriate learning rate is fairly difficult. In earlier works, the learning rate of 1D-LSTM was normally set to a value between 10^{-4} and 10^{-5} [Gra12], and for MD-LSTM between 10^{-5} and 10^{-6} [GFS07b]. In this thesis, a small subset of the training or validation set was preliminarily tested with a range of constant values between 10^{-3} and 10^{-7} . The results of these tests showed that as the learning rate decreases, error changes become more stable over the iterations, especially when using a large sized input for segmentation. Note that, there is another possibility to set an adaptive learning rate which allows to change the learning rate during the training process to provide stable learning. For the final experiments, the learning rate has been fixed to either 10^{-4} or 10^{-6} , depending on the task and the size of input. *Momentum* is typically set to 0.9, which remains constant throughout the experiments conducted in the scope of this thesis.

4.4 Conclusion

This chapter described the details of LSTM networks for 2D and 3D image analysis. The details of the network architectures and network settings were explained. In the following three chapters, the network architectures introduced here will be evaluated on various types of images and tasks: image classification on 2D texture and scene images, image segmentation on 2D texture and scene images, as well as on 3D biomedical volumetric images. The next chapter will start with the experiments of 2D-LSTM for image classification on texture and scene images.

Chapter 5

Image Classification

In this chapter, the 2D-LSTM model presented in the last chapter has been evaluated on various types of images: texture and scene images. Here, an additional layer, called a *collapse layer* is added between the hidden layer and the output layer to combine all output activations of input pixels. These accumulated activations contribute to the final class label of a given input image. The network for classification consists of four layers: an input layer, a hidden layer with LSTM, a collapse layer, and an output layer (see Section 4.1 and Section 4.2 for more details).

Texture images are firstly applied to see how well LSTM-based image classification works compared to other approaches. Here, several ways of feeding the input into the network and the output integration technique are presented. The input is represented as multi-patches with a variety of transformations: scaling, rotation, and translation. The patch-wise input can easily be combined with such transformations providing flexibility of input representation and invariance in the network. Compared to using a 2D image as a whole, multi-patch input with the output integration has the flexibility to represent the pixels to the wide range of scaled and rotated textures which produce consistently better results on five widely used datasets for texture classification.

As for more realistic scenarios, this approach is subsequently applied to *natural scene images*, collected from web-searches. Due to the lack of availability of a large and realistic data source, a new *web-scene image dataset* is created. This dataset contains different types of images for training and testing. For training, images contain a single visual content per image (e.g., sky, grass, and building). It makes training easier as compared to using complex and noisy scene images directly. For testing

purposes, natural scene images are randomly collected from the web, which contain many unrelated contents and noise. With help of several pruning rules, the images are classified with multiple labels under such conditions. The experiments on the web-scene image dataset show that LSTM-based classification outperforms other methods including CNN which is one of the most popular approaches in the field. The approach yields good results on automatic web-image tagging, which is the real-world application using this approach. Further experiments on the recently published dataset, *outdoor scene attribute dataset* reports a significant improvement over the baseline [WJWZ13] and CNN (the improvement ca. 21% compared to the best accuracy of other approaches).

The work presented in this chapter appeared in ICPR 2014 [BLB14] and Pattern Recognition Letter 2015 [BLB15]. Section 5.1 describes the details of an approach for texture classification and its experiments. Section 5.2 presents the attribute learning with LSTM on natural scene images and shows the experimental results on multi-label scene understanding and its application, web-image tagging.

5.1 Texture Classification

The main purpose of this work is to examine how well LSTM-based texture classification works compared to other approaches. A standard 2D-LSTM is directly applied to raw RGB value of pixels, without any manually designed feature extraction or pre-processing. Here, multi-patch input with various transformations¹ and the output integration technique increased the quality of the performance and reliability of the networks for classification. The complete flow diagram of the approach is shown in Figure 5.1.

5.1.1 The Approach

Training: The network receives input from raw RGB pixel values. In general, the training data in the datasets have a small number of samples and are under fixed conditions; they are under small changes in pose, scale, and illumination, and the samples are not diverse enough to generalize the network. To achieve robust training,

¹i.e., scaling, rotation, and translation

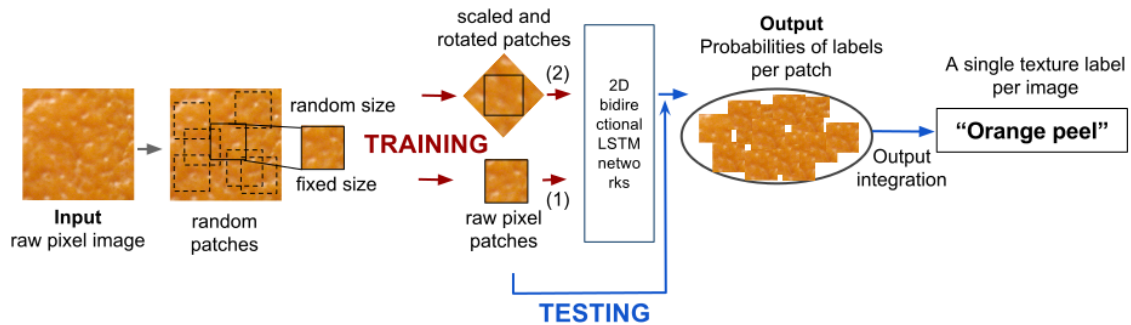


Figure 5.1: A pipeline for texture classification. The input image is the raw RGB pixel values. For training, the multi-patch input with diverse transformations are considered (Both with transformations in (2) and without transformations in (1) on the training input data are evaluated in the experiment.) To apply the various transformations, an amount of random patches are first sampled. These multi-patches are randomly rotated and scaled then sent to the networks. Finally, predicted outputs are integrated to decide the final output; the maximum score of class probabilities and its corresponding label are selected as a final output of the image.

several transformations like rotation, scaling, and translation are applied on multi-patches sampled from an input image. The network is primarily trained on (1) the original patches, (2) transformed patches, (3) an image as whole. In Section 5.1.4, the performance of these are all evaluated. To transform an input image, randomly sized patches are first selected at a random position. They are then rotated and scaled-up or down into the size of $n \times n$. This process is repeated multiple times for each input image. It allows us to train on randomly scaled and rotated samples as well as on the original patches, which can easily be applied to the network without prior knowledges of the image resolution and condition. This way, our model can capture the variation of each attribute under limited number of training samples and keep the input dimension constant to retain one optimal model for different data.

Output integration: The networks are expected to output conditional probabilities of all labels given each patch j : $Pr(\text{label} | \text{patch}_j)$. To determine the final class of an image, further integration process is required. Since we sampled the random parts of an image, some patches have higher distinctive patterns and some may contain noise or ambiguous patterns. For this reason, all output from the network (the class distributions) are smoothed before deciding the final prediction that find the most probable label of the image. The output vectors over all patches are first averaged, and then the label with the maximum score over the vector becomes the

Table 5.1: The summary of texture datasets used in the experiments. Each database has various challenging problems for separate experimental designs. Textures in each dataset are taken under a variety of conditions, quality, and resolutions. KTH-TIPS includes specific texture from materials under varying illumination, poses, and scales. Other datasets are from natural textures of a scene or an object. OuTex, VisTexL, and VisTexP contain a variety of textures with small number of training images. In contrast, NewbarkTex is composed of six tree bark classes with a larger number of training images. However, variation between classes is not distinctive. Difficulties on these datasets are shown in Figure 1.3.

	Image size	# Texture	# Training per class	# Test	Type of texture
KTH-TIPS [KTH]	200×200	10	40	410	material
OuTex [out] (OuTex-TC-00013)	128×128	68	10	680	natural texture
VisTexL [visa] (Contrib-TC-00006)	128×128	54	8	432	natural texture
VisTexP [visb]	128×128	55	8	440	natural texture
NewbarkTex [bar]	64×64	6	136	816	natural texture

final decision:

$$\arg \max_{\text{label}} \frac{1}{\# \text{ patches}} \sum_{j=1}^{\# \text{ patches}} Pr(\text{label} | \text{patch}_j) \quad (5.1)$$

5.1.2 Datasets

The dataset *KTH-TIPS* [KTH] includes various conditions, namely nine scales spanning two octaves, three different illumination directions, and three different poses. Some materials have very similar textures like cotton and linen or sponge and brown bread, which make the database challenging. For comparison, the evaluation setup proposed by Zhang et al. [ZMLS07] has been followed.

The dataset *OuTex* [out] contains 68 classes of various color textures with 128×128 sized image. Half of the images for training (680 images out of 1360 images) and the remaining is used for testing. Several image categories have similar color and texture, so discriminating only by their pixel values is not trivial.

The next datasets *VisTexL* [visa] and *VisTexP* [visb] are both designed for natural color textures under non static conditions. The same scheme is used to generate the dataset. For *VisTexL*, 864 disjoint sub-images are generated from 54 texture images. *VisTexP* includes 55 texture classes with 880 sub-images. For both datasets, each image (size 512×512) is split into 16 sub-images (size 128×128). These sub-images are considered as the same class. As for the *OuTex* set, half of the images are used in the training phase.

Recently, a new benchmark color texture image test suite, *NewbarkTex* [bar] is proposed using a subset of the *BarkTex* dataset [MVK⁺02, Pal04, PL02, PVM07]. Six tree bark classes with 68 images per class (128×128) are divided into 4 sub-images (size 64×64). A total of 272 sub-images per classes (total 1,632 images) are built which are again divided into halves for training and testing.

5.1.3 Experimental Setup

All the experiments have been performed by using the RNNLIB library². For the statistical evaluation proposed by Flexer [Fle96], a preliminary test is repeated five times with numerous parameters to find the appropriate network architecture. The optimal parameters are then applied to the datasets with randomly divided training and testing samples. It is repeated 50 times and reported the average accuracy. All five datasets have been evaluated directly on the raw RGB pixel values.

Input representation: As mentioned in Section 5.1.1, a wide range of scale and rotation are considered as input. For rescaling the input images, patches between 50×50 and 80×80 are randomly sampled, then resized to 64×64 pixels. The scaled patches are then rotated at angles between 0° - 360° . Both scale and rotation are of 1 pixel or 1° increment. Besides, the number of patches extracted in an image also affects the performance since randomly rotated and scaled patches increase diversity. Very small and large number of patches (10 and 200) have been examined for all experiments to evaluate the influence of performance.

Input block and LSTM networks: To find an optimal network model with proper input size and its corresponding input block, the range of parameters (hidden

²<http://sourceforge.net/projects/rnnl/>

size = {15, 25, 50, 75, 100}, input block size = {non-block (pixel-wise), 5, 10, 15, 20, 25} with the input pixels = {64 × 64, 100 × 100, 200 × 200}) has been preliminarily examined. If no input block is used, the block is considered as one pixel (pixel-wise). An important finding from this experiment is that training became more difficult as the length of input increases. It is mainly due to the long-term dependency issue of LSTM mentioned in Section 3.1.3 and Section 4.1. The proper training has been started when the size of the input block is bigger than 5 × 5 if the input image has more than 50 × 50 pixels. In other words, MD-LSTM cannot learn when the length of input is longer than 2500 pixels, but the total length of the input became shorter by using the input block. Thus, the network can avoid the difficulty of learning long-term dependencies in the long length input. In addition, the MD-LSTM network did not converge if the size of input block is small but the number of hidden units is big; pixel-level (block size 1 × 1) or block size 5 × 5 when a hidden size is bigger than 25. These preliminary experiments have shown the influence of input block and relationship of input and hidden size with block size. At the end, the input block size of 5 with 15 hidden unit was set with a input size of 64 × 64 pixels for all experiments. The learning rate and momentum have been fixed for all experiments to 1e-4 and 0.9, respectively.

Evaluation: The performance is evaluated using per-patch and per-image accuracy in order to compare the effectiveness of a patch-based input representation. Given input = { i_1, i_2, \dots, i_N } with corresponding labels = { l_1, l_2, \dots, l_N }, the patch = { $p_1, p_2, \dots, p_M, \dots, p_T$ } are samples from N images. N is the number of images, M is the number of patches per image, and $T = M \times N$. The classification accuracy per-patch is computed as follows:

$$\text{accuracy}_{\text{per-patch}} = \frac{1}{T} \sum_{t=1}^T \begin{cases} 1 & \arg \max_l Pr(l|p_t) = l_g, \\ 0 & \text{otherwise,} \end{cases} \quad (5.2)$$

where p , l , and l_g indicate the input patch, its predicted label, and the GT label, respectively.

Furthermore, per-image accuracy is measured using an integrated score of all patches:

$$\text{accuracy}_{\text{per-image}} = \frac{1}{N} \sum_{n=1}^N \begin{cases} 1 & \arg \max_l \frac{1}{M} \sum_{m=1}^M Pr(l|p_m) = l_g, \\ 0 & \text{otherwise,} \end{cases} \quad (5.3)$$

Table 5.2: Correct classification rates (avg. accuracy, %) on five benchmark datasets of texture classification. The experiments of all datasets with the best resulted parameters are repeated 50 times for the statical evaluation and the averaged accuracies are reported (the size of input patch = 64×64 , the size of input block = 5×5 , the number of hidden units = 15, the learning rate = $1e-4$). The performance is compared to the state-of-the-art methods for texture classification. The LSTM networks lead to superior performance on most datasets. Note that the performance in bold is statistically significant with 95% confidence among other methods, and underlined numbers indicate comparable results. Baseline methods: Basic Image Features based on steerable filters (**BIF**), Multi-scale Local Binary Patterns (**LBP**), Principal Curvatures with four scales (**PC**), Rotation invariant multi-scale features (**MLEP**), Semi-joint Texton descriptor (**STD**), Homogeneous texture+color structure (**HTD+CSD**), Multispectral co-occurrence (**MM**), and Haralick from reduced size chromatic co-occurrence (**RSCCMs**)

Dataset	KTH-TIPS	OuTex	VisTexL	VisTexP	NewbarkTex
# test samples	610	680	432	440	816
BIF [CG10]	98.50	-	-	-	-
LBP [ZLZ13]	93.17	-	-	-	-
PC [ZZL12]	97.52	-	-	-	-
MLEP [ZLZ13]	96.41	-	-	-	-
STD [AV12]	-	90.32	99.25	98.89	-
HTD+CSD [MOVY01]	-	86.71	<u>99.56</u>	98.53	-
MM [ADBB04]	-	94.1	-	97.9	-
RSCCMs [PVM10]	-	-	-	-	75.9
LSTM networks	100	<u>94.70</u>	99.09	<u>99.07</u>	78.2

5.1.4 Results and Analysis

The five tested datasets are fed in three different ways: (1) a 2D image as whole, (2) multi-patches without any transformations, (3) multi-patches with transformations in scale and rotation. Note that the results reported in Table 5.2 are with input type (3), which have gotten the best accuracy among them. Here, the performance of all of these input are compared and analyzed. With input type (1), KTH-TIPS and NewbarkTex have already been reached the best accuracy among the feature extraction based approaches (99.48% and 78.2% respectively) and others are comparable (93.09% for OuTex, 89.55% for VisTexL and 90.0% for VisTexP). With multi-patch input (input type (2) and (3)), per-image accuracy are much more scattered than when using per-patch (the difference was about 3%). The number

of patches per image has also an important role on classification performance. The performance with more number of patches (maximum 200 in our experiment) per image is higher on most datasets (around 2% higher for all datasets except OuTex) than other approaches. The best results using LSTM networks compared with different feature extraction based methods are summarized in Table 5.2. Overall, the best accuracy of our approach led to the superior performance on most of the benchmark datasets. Specifically, 200 patches per-image accuracy of KTH-TIPS dataset have achieved 100% (1.5% higher) and NewbarkTex dataset achieved 78.2% (2.3% higher). The Statistical significance is lower for the dataset OuTex, VisTexL and VisTexP because of extremely small number of training samples with a large number of textures (only 10 images per class in OuTex (68 textures) and 8 images per class in VisTexL (54 textures) and VisTexP (55 textures)). However, it still gives comparable performance. The results show that LSTM combined with the multi-patch input with various transformations is very powerful to discriminate the raw pixel level images.

5.1.5 Summary

This section explored the capacity and efficiency of 2D-LSTM for image classification. The network is examined on texture images and show a high performance gain on five commonly used texture classification datasets. In the following section, the system handles natural scene images that may contain an amount of clutter, noise as well as some ambiguous contents in an image.

5.2 Scene Understanding

The task of scene understanding is to classify all containing semantics from a scene. Unlike object recognition, scenes cannot be identified by a single category; each scene contains multiple salient attributes, for instance, sky, buildings, and ground. Recently, learning higher-level semantic contents, such as object parts and materials, beyond the plain visual cues, such as color and texture, has achieved significant performance gains [FZ07, MLKS13] in object recognition. Inspired by the idea of learning semantic attribute, *mid-level attribute learning* is introduced in this section to describe scene images. The mid-level attributes in scenes are visual concepts, which are closely related to the visual properties, such as sky, ocean, or sand. The learning process is similar to texture/material classification as described in the previous section. However, in this section, networks are trained on mid-level attributes, which are part of a scene and appear in complex scenes along many other elements. They are then evaluated on natural scenes containing a mix of mid-level attributes collected from web-searches. The diagram of the approach is illustrated in Figure 5.2.

5.2.1 The Approach

Mid-level Attribute Learning: An LSTM network model takes mid-level attribute images containing a single attribute per-image, like sky, ocean, or building. Some examples of mid-level attribute images are shown in Figure 5.2-left. The multi-patches with transformations in scale and rotation (input type (3) explained in Section 5.1.1) are also applied for attribute learning.

Visual Attribute Classification: After training, models are first evaluated on single-attribute images similar to the training data (Experiment 1; Figure 5.2, left). During the training phase, a number of patches are randomly extracted from a single image and passed through the network. Like the texture classification task, the outputs of given patches from the networks are then integrated into one attribute label. Due to the way the dataset is generated, images contain irrelevant (noise, clutter, or watermark) and mislabeled regions. We use the following smoothing process to correct these. As in the input integration step (Equation 5.1), the best label

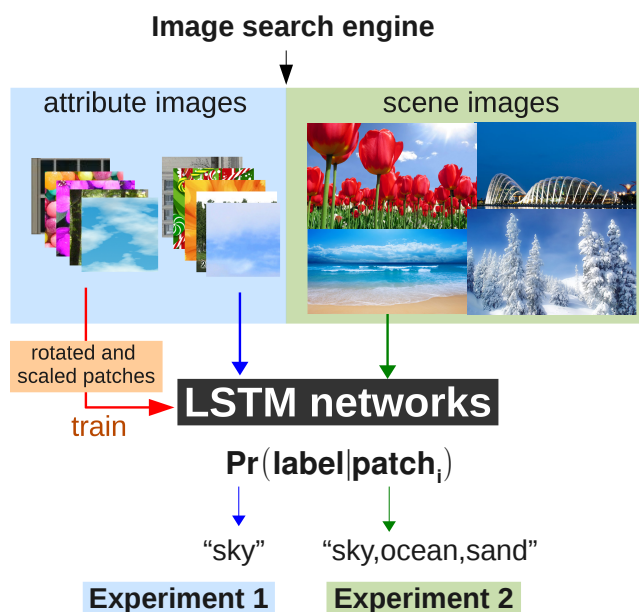


Figure 5.2: An overview of the scene analysis system. *Training:* Mid-level attribute learning (red). 2D LSTM recurrent neural network model is trained on the mid-level visual attribute data collected from a web image search engine. *Experiment 1:* Visual attribute classification (blue). It is firstly tested on single attribute images. The images contain only a single attribute per image (e.g., “sky”) *Experiment 2:* Natural scene analysis (green). Real-world scene images are used for this experiment. These images include several attributes, e.g., “sky, ocean, and sand”.

is determined by the average of all of output scores. To evaluate the performance, per-image accuracy (Equation 5.3) is measured using an integrated score (Equation 5.1).

Natural Scene Analysis: Although the networks are trained only on specially constructed data (attribute images), they are evaluated on natural images obtained from the web-searches containing a mix of mid-level features (Experiment 2; Figure 5.2, right). Like the visual attribute classification task, multi-patch inputs are sent to the network model, and each patch is classified according to its mid-level attribute. The natural images obtained from the web-searches contain a wide range of scales, resolutions, rotation, and clutter. The following experiments are intended to demonstrate that our method works on such images and can be directly applicable to more realistic applications. Here, to handle the scene images containing multiple attributes, two pruning rules, *probabilistic patch pruning* and *top-k rank pruning*, are introduced.

The random multi-patches may contain noise or unrelated contents, such as logos, watermarks, or ambiguous attribute textures. To avoid these issues, two pruning

rules are introduced. Such noisy patches are first pruned using *Probabilistic patch pruning* to avoid the confusion on the final decision. This pruning rule is based on the posterior class probability $Pr(l|p)$ of the label l given the patch p and a threshold T_p :

$$\max_l Pr(l|p) > T_p \quad (0 < T_p \leq 1) \quad (5.4)$$

Another pruning rule for robust classification is *Top-k rank pruning*. The basic observation for this rule is that the highly probable visual patterns, which tend to recur frequently, are the main semantic regions of the scene. For the remaining patches after probabilistic patch pruning, the ranking score is computed from the output integration (Equation 5.1), and the potential attributes are then ranked based on their ranking score. Since the number of semantic regions (the number of k) is uncertain, we cannot easily define an optimal k . To handle this problem, top-k ranked list passes the threshold T_r :

$$\frac{1}{S} \sum_{s=1}^S Pr(l|p_s) > T_r \quad (0 < T_r \leq 1), \quad S > N_p, \quad (5.5)$$

where S is the number of patches with a corresponding label after the patch pruning. N_p is a constant integer value for pruning predicted labels that are not stable. The label is also rejected if the final number of patches of the label is less than N_p . Thus, it prunes unreliable class labels from a statistical observation (the frequencies of the highly probable patches of the label) on the image.

5.2.2 Datasets

(Proposed) Web-Scene Image Dataset

Google's image search³ is used for collecting both the attribute and the scene images. Twelve common mid-level categories are chosen: some frequent and generic in outdoor scenes (building, flower, forest, grass, snow, ocean, sand, sky, and gravel), and narrow (stucco, candy, and meat). The aim of constructing this dataset is to achieve the following properties: 1) plenty of diverse samples are created, 2) the uncontrolled raw web-data containing noise and errors are directly applied to the algorithm without manual annotation or pre-selection, 3) randomly collected scene images (unseen

³<http://images.google.com>

and unknown data) represent the real-world data and the understanding of them is only by the visual attribute information, and 4) web-images retrieved from a web image search are used directly for tagging, which represent a more realistic scenario.

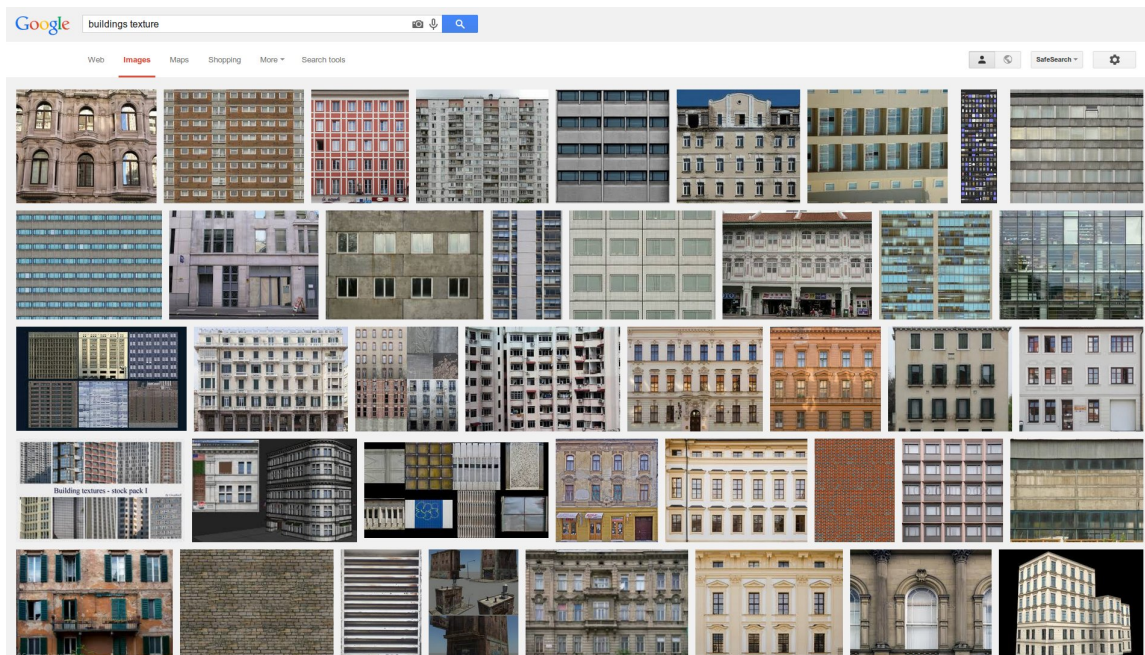
Attribute data: Ideal training data for this task would consist of manually segmented and labeled natural images, as for example, found in the databases [GFK09b, WJWZ13]. However, little training data of that form is available. Therefore, the training data is created using a web-search engine querying the attribute-texture, “*keyword* + texture”, e.g., “building texture”. The images are with mostly well oriented visual attribute patterns (see Figure 5.3).

From the search, around 95 images for each class with different sizes have been obtained. The collection of images is then split into 8-24 disjoint sub-images that are used for mid-level attribute representation. Note that, we follow the same data generation procedure as other common texture classification datasets⁴. The dataset is divided into training (350 images per class), testing (196 images per class), and validation (50 images per class). The training and validation sets are used for texture learning and the learned model is tested using the test set. Some examples of the visual attribute dataset from the web are shown in Figure 5.4. Here, no manual correction or validation is considered from the collected images, so the training data contains noisy label⁵ and other artifacts (see Figure 1.4). The experiments show that the LSTM networks trained on such noisy data can be immediately applicable on the web. Around 600 attribute images per class are generated and are split by assigning 60% to the training set, 10% to the validation set, and 30% to the test set.

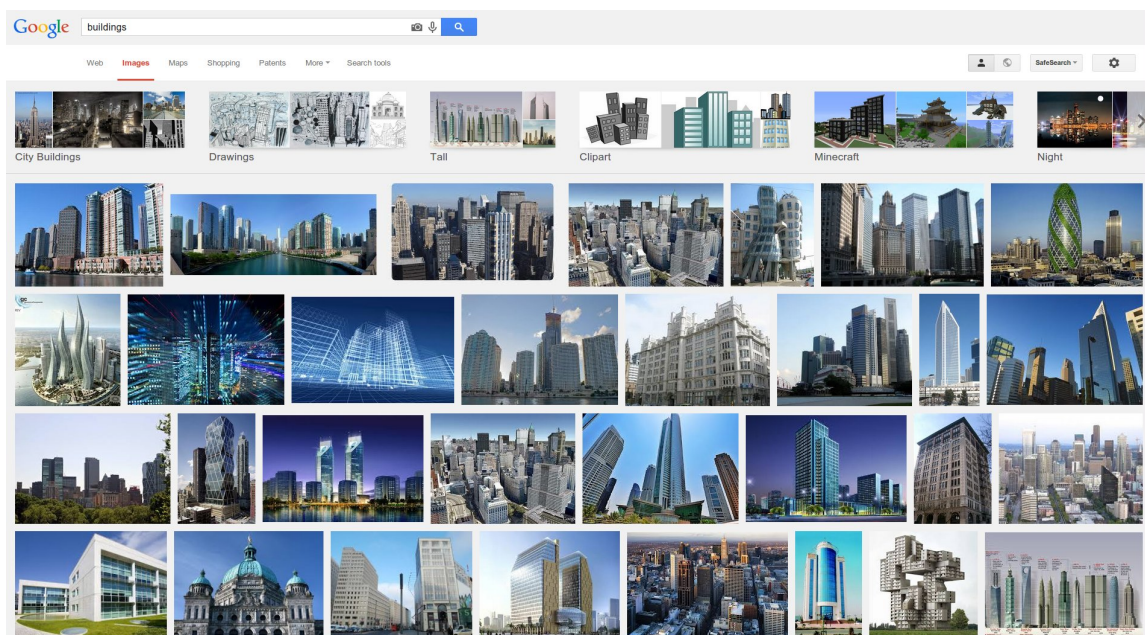
Scene data: Natural scene images contain a mix of mid-level attributes. Scene images are randomly collected from the web-search using “*keyword*” as query, e.g., “building”. The only data-cleaning that has been performed is to discard duplicated images in the results; there is also no overlap between scene and visual attribute data. This means that for each category, a wide range of representations of that category may occur. They may contain multiple (most likely including the keyword), single, or not related visual attributes which images are more challenging (see Figure 5.5).

⁴e.g., KTH-TIPS and KTH-TIPS2 texture image database (<http://www.nada.kth.se/cvap/databases/kth-tips/download.html>)

⁵Web-search engine may retrieve wrong output from the query



(a) visual attribute image (keyword: “building texture”)



(b) scene image (keyword: “building”)

Figure 5.3: Image search for database generation: Images returned from Google’s image search for the class “building” ((a) visual attribute data, (b) scene data). The collected images may have well-conditioned natural images but also lower quality images: drawings or cartoon of the keyword, watermarks or logos, extensive noise, background or irrelevant parts, low-resolution texture or some other fault. However, neither the training nor the test sets have passed through any post-selection or manual annotation. Using randomly collected images from the Internet, the experiments show the robustness of our approach which itself handles these issues.

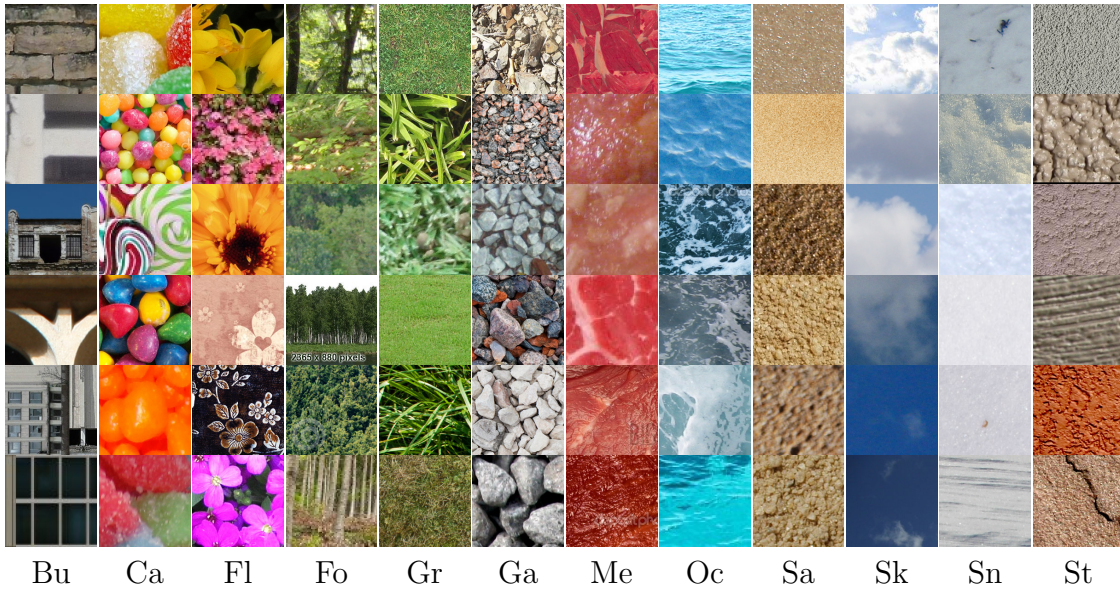


Figure 5.4: Examples of mid-level attribute data. The selected keywords: building(Bu), candy(Ca), flower(Fl), forest(Fo), grass(gr), gravel(ga), meat(Me), ocean(Oc), sand(Sa), sky(Sk), snow(Sn), and stucco(St).

For example, images with the query “stucco” contain a building or a person applying the stucco on a building. In addition, web-images contain watermarks, logos, or are low-resolution images which have been also included in this scene data. Some examples of the scene images, their keyword, and visually recognizable attributes in the scene, are shown in Figure 5.5. This data is made only for the test phases, i.e., scene analysis and web-image tagging. 540 scene images containing multiple classes in each image are used for estimating scene attributes.

Outdoor Scene Attribute (SceneAtt) Dataset

The approach is also evaluated on the public scene dataset (SceneAtt) proposed by [WJWZ13]. This dataset is selected since this study is the most similar to our work. As reported in [WJWZ13], most of the public scene datasets focused on the specific objects, humans or the functional activities; in contrast, our goal is to analyze the all visible contents of the scene. Furthermore, this dataset contains precise text descriptions with weak labels (not precise), which makes the experiment more complex and realistic.

The dataset is collected from LMO [LYT09], SUN attribute dataset [PH12], Google images, and Flickr. It consists of 1226 images of 256×256 pixels and 30 noun + adjective attribute pairs. The dataset is split into 645 images for training and the



Figure 5.5: Examples of scene data for scene analysis and web-image tagging. The selected keywords: building, candy, meat, flower, forest, grass, snow, ocean, sand, sky, gravel, and stucco (same as visual attribute classification task). The searched query on web image search is shown above the images, and visually recognizable attributes are listed below each image. The scene image normally contains multiple visual attribute parts. For instance, the top first image with the query “building” includes sky, ocean, and building parts.

rest for testing.

5.2.3 Experimental Setup

Web-Scene Image Dataset: Two separate experiments are performed to evaluate and compare the performance of our approach. The first experiment evaluates the quality of mid-level attribute learning using the attribute data (containing a single attribute). The second part considers a more realistic scenario; the attribute regions in a scene image are analyzed using the learned model, and the best tags are extracted using the pruning rules.

For training, 200 patches from an input image are randomly sampled with a size between 50×50 and 80×80 . The patch is then rotated at angles of $0^\circ - 360^\circ$ and rescaled to 64×64 . Both scale and rotation are with 1 pixel or 1° level increment. For testing on scene images, 200 patches are collected for final top-k prediction. A fixed number of patches are extracted for all images, despite the huge variations in resolution. This shows the robustness of the approach under a variety of images. The threshold of the pruning rules, T_p , T_r , and N_p are selected empirically to 0.6, 0.4, and 10, respectively. The same parameters are kept for all experiments.

For LSTM training, the RNNLIB library⁶ is used. For all experiments, the size of the input block, hidden size, and learning rate are fixed in 5, 15, and 1e-4 respectively.

As a baseline, the performance is compared to feature-based and filter-based methods:

- (feature-based) C-PHOW-BoVW: PHOW features on the three HSV image channels and BoVW [BZM07b]. The PHOW feature is a variant of dense-SIFT descriptors, extracted at multiple scales.
- (feature-based) C-PHOW-FV: PHOW features on the three HSV image channels and Fisher Vector (FV) [PD07, SP11].
- (feature-based) C-DSIFT-FV: Dense-SIFT features on the three HSV image channels and FV.
- (filter-based) C-Gabor: Gabor with color chromatic features.

⁶<http://sourceforge.net/projects/rnnl/>

- (filter-based) C-Co-occurrence: Co-occurrence with color chromatic features.

As features, standard dense-SIFT and PHOW feature are used. After, multiple encoding schemes including BoVW and FV are applied on the extracted feature. The detailed feature encoding steps are as follows: Firstly, a dense feature extractor is applied on input images. The features are then clustered using k-means, then vector quantization is performed using either a kd-tree (for BoVW) or a Gaussian mixture model (for FV). Finally, visual words are accumulated into histograms with a spatial pyramid encoding. 80 dimensions, 1024 words and 64 words are used for PCA, BoVW, and FV, respectively. The open library VLFeat [VF10] has been used for all feature extraction and classification methods.

For the comparison with filter-based approaches, the best low-level features reported in [BHSF11b], Co-occurrence and Gabor with color chromatic features [DW01, ADBB04, BHSF11b], are selected. Eight co-occurrence matrices corresponding to one-pixel displacements along the following eight directions: $\{0^\circ, 45^\circ, \dots, 315^\circ\}$ are used. Five statistical features, namely contrast, correlation, energy, entropy, and homogeneity, are extracted in each direction, and averaged for rotation invariance. These features are normalized between 0 and 1. For Gabor filter, a bank of filters with the following parameters is used: number of frequencies = 4, number of orientations = 6, maximum frequency = 0.327, frequency ratio = half-octave. All parameters are set based on the work from [BHSF11b]. All features are extracted in HSV color space and SVMs with Chi Squared kernels of period 2 have been used as classifiers. All above parameters are optimized empirically.

For CNN experiments, the Caffe library [Jia13] is used. Hyper parameter search is carried out based on the work of [KSH12a]. The optimal structure identified by this process consists of five convolutional and two fully-connected layers with half the size of dimensions of the Krizhevsky's network architecture [KSH12a]. It is observed from the experiments that using fewer than five layers result in significant decreases in recognition performance, e.g., 95.79% with five convolutional and two fully-connected layers, and 90.09% with four convolutional and one fully-connected layer. Pre-training on ImageNet for the network initialization for NN-based methods is often effective [GDDM14b], but it is not used for both LSTM networks and CNNs in these comparisons.

Table 5.3: Accuracy comparisons of single visual attribute classification on web-image dataset (The best score is shown in bold). In order to compare the performance, all of our experiments have been following the same experimental setup. *Visual Attribute Classification (single attribute classification)*: The best accuracy of our approach lead to superior performance compared to other common approaches. (No. test samples = 2352, 95% confidence interval = ± 0.20).

Method	Attribute	# weights for NNs
	Accuracy (%)	
C-PHOW-BoVW, SVM [BZM07b]	59.86	-
C-PHOW-FV, SVM [SP11]	68.28	-
C-DSIFT-FV, SVM	72.66	-
C-Gabor, SVM [BHSF11b]	62.12	-
C-Co-occurrence, SVM [BHSF11b]	75.21	-
CNNs [KSH12a]	95.79	38,802,300
LSTM networks	97.32	100,272

C-: HSV color space

SceneAtt Dataset: CNNs and the LSTM networks are trained on 645 training images with the same parameter setting as the web images from our database. This experiment is intended as a harder test case because of more attributes, different descriptions, and small training samples. It means that it is not appropriate to use the model trained from the previous experiment (on web-scene image dataset) for this dataset. Thus, a new LSTM model is trained and evaluated using the same training and test data as Wang’s work [WJWZ13].

5.2.4 Results and Analysis

Web-image Dataset: The first experiment on single attribute images shows that our approach outperforms other baseline methods and CNNs (Table 5.3). As pointed out by [MS05b], many popular methods are limited to the specific types of texture features. For instance, PHOW-BoW performs well only on repetitive-textures and some structured-textures, e.g., gravel and building. In addition, Co-occurrence or Gabor feature discriminates well on limited color-texture image datasets [BHSF11b], but suffers from various attribute types and its diversity. As the worst case, if there is

Table 5.4: Accuracy comparisons for natural scene analysis on the web-image dataset (The best score is shown in bold). In order to compare the performance, all of our experiments have been following the same experimental setup including the number of weights. *Natural scene Analysis:* multi-attribute classification; since we cannot directly make one decision of a classifier for scene analysis (multiple visual attributes), top-k ranked list is used to decide the highest probable visual attribute classes of a scene. The visual attributes are listed based on its ranking score in descending order, and top-3 and top-5 accuracies are considered to compare the performance of our approach with other methods (No. test samples = 540, 95% confidence interval = ± 0.42).

Method	Scene		
	Top-1 (%)	Top-3 (%)	Top-5 (%)
C-PHOW-BoVW, SVM [BZM07b]	8.56	31.91	48.84
C-PHOW-FV, SVM [SP11]	8.38	35.12	53.30
C-DSIFT-FV, SVM	9.09	32.98	50.62
C-Gabor, SVM [BHSF11b]	42.34	70.88	81.03
C-Co-occurrence, SVM [BHSF11b]	46.17	74.52	82.38
CNNs [KSH12a]	56.81	79.85	90.21
LSTM networks	71.43	84.23	94.25

C-: HSV color space

not enough textures on a given image, the algorithm cannot extract sufficient number of features whichever feature detectors or descriptors are chosen. However, the results from 2D-LSTM networks outperform other approaches under various attribute types and transformations, including the low-textured attribute and huge distortions, without any hand-designed features.

Here, the real-world application, *automatic web-image tagging*, is further demonstrated using mid-attribute learning and the proposed web-scene image dataset. The quality of scene analysis is evaluated using top-k ranked list (Table 5.4). It predicts the top-k most relevant attributes. k is the maximum number of attributes (tags) in the scene to be predicted and the ranking score indicates highly probable attributes. The confusion table ($k = 1$) is shown in Figure 5.6; the major portion of keywords is predicted as top-1. The scene images include multiple attributes, where a major portion does not always correspond to the keyword — the weakness of the web image search engine. Therefore, a considerable number of images in each class are predicted as sky, and it shows that the system can potentially improve the image

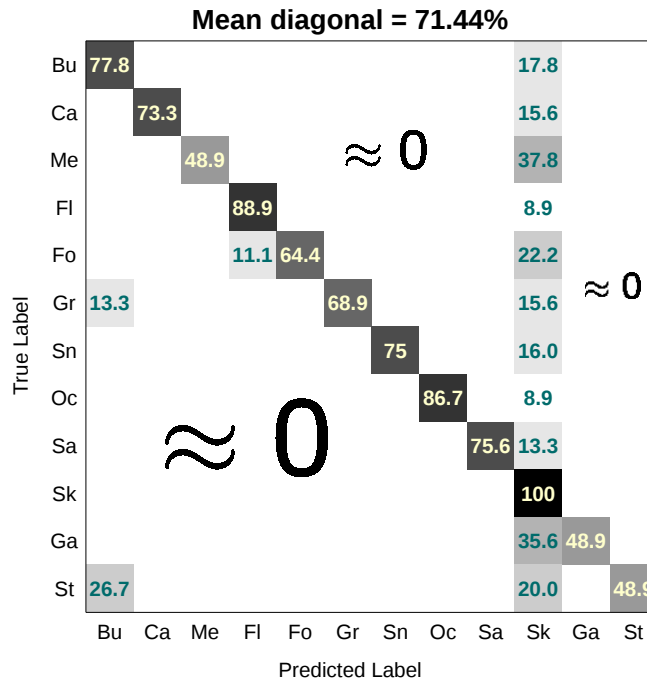


Figure 5.6: Confusion table with top-1 predicted semantic attribute on scene images. The column is the top-1 predicted semantic attribute and the row is the GT label (labels from top to bottom (column) and left to right (row): building(Bu), candy(Ca), meat(Me), flower(Fl), forest(Fo), grass(Gr), snow(Sn), ocean(Oc), sand(Sa), sky(Sk), gravel(Ga), stucco(St)).

search engine. Especially, the query “stucco” is predicted as building in some images, since the stucco could be a part of the building depending on the scale. In this case, building-like images are retrieved from the web-search engine. However, LSTM networks predict them as building on the top-1 list (see the last column in Figure 5.7; keyword: stucco, tagging result: building, sky, and gravel).

Since most scene images in the dataset contain around one to five semantic classes (mostly up to three), top-5 lists are most likely to provide all semantic parts. For performance evaluation, top-1, top-3, and top-5 results are compared with their keyword (The keyword is considered as a GT label). In addition, automatic tagging results of each image are shown; Figure 5.7 shows examples of the tagging results and one can see that they are correctly predicted by using their associated top-ranked list.

SceneAtt Dataset: In Table 5.5, Mean Average Precision (MAP) scores of LSTM and CNNs are compared with the methods reported in [WJWZ13]. The best method in [WJWZ13], HST-att learns the spatial layout and attribute association by the

Table 5.5: The comparison of Mean Average Precision (mAP) on SceneAtt dataset.

Method	MAP(%)
eKernel+SVM [XHE ⁺ 10]	64.48
BoW+SPM [LSP06]	53.11
HST-geo [WWZ13]	51.67
HST-att [WJWZ13]	67.58
CNNs [KSH12a]	63.24
LSTM networks	88.59

scene’s appearance model. It then finds the most probable parse tree of the adjective and noun description. To compare with CNNs, the same architecture is trained (without pre-training) as in the previous experiments. Using a simple LSTM network, MAP went about 21% higher than HST-att, and 25% higher than CNNs. The performance of CNNs can further be improved by using the pre-trained model on ImageNet as mentioned earlier, since the training data is scarce.

5.2.5 Summary

This section presents ways of resolving the issues of natural web-scene images. These web-images are extremely noisy and contain numerous unrelated visual content. Furthermore, an unknown number of visual content under these conditions makes the task challenging. 2D-LSTM networks with multi-patch pruning rules show a performance gain with a large margin on both Web-scene image and SceneAtt datasets. Web-image tagging application illustrates that this approach can be applied to a real-world scenario.

5.3 Conclusion

In this chapter, a 2D-LSTM network model for image classification has been described. MD-LSTM with an activation integration layer (collapse layer) results image classification without any extra steps. Here, a simple network architecture works well compared to non-neural network methods, and that among neural networks,

2D-LSTM networks outperform deep CNNs with only a small number of hidden units compared to CNNs. The experiments show that data augmentation, multi-patch integration, and pruning rules combined with LSTM networks can deal with several types of images: texture/material images and randomly collected scene images.

We now turn to a more complex underlying problem in computer vision, i.e., image segmentation in the next chapter. By taking advantage of LSTM's unique characteristics, pixel-level classification is introduced in an end-to-end manner.






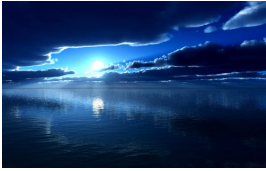



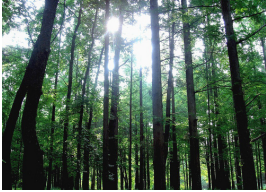





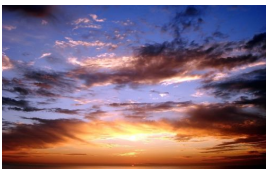
image	keyword tagging result	image	keyword tagging result
	building sky, building, ocean		ocean sky, sand, ocean
	grass sky, grass		sand sky, sand
	sky sky		sky ocean, sky
	ocean sky, ocean		ocean ocean, sky
	forest forest, sky, grass		forest forest, sky
	snow sky, snow		ocean sky, ocean
	grass sky, forest		grass snow, grass
	sky sky, ocean		sky sky

image	keyword tagging result	image	keyword tagging result
	candy candy		candy candy
	building sky, building, ocean		building building, sky
	snow sky, snow		ocean sky, sand, ocean
	ocean sky, ocean		ocean ocean, sky
	forest forest, grass, sky		stucco stucco
	stucco building, sky, gravel		ocean sky, ocean

Figure 5.7: The results of automatic web-image tagging. Top-3 after the patch pruning is used as tags of each image (The order of the list indicates a higher rank). As it can be seen from tagging results, relevant attributes are well-detected for each scene. Our system can even improve upon the retrieval system. For instance, the wrongly retrieved image from a web-search engine (e.g., the left bottom image — keyword: stucco, correct semantic attributes: building, sky) can be corrected by our tagging system.

Chapter 6

Image Segmentation

This chapter presents 2D image segmentation (pixel-level classification) using MD-LSTM network architecture as described in Section 4.1 and Section 4.2. As mentioned in Section 1.1.1, many segmentation methods mainly consider a small local context around each pixel for segmentation. The main issue in segmentation is to integrate a large input context into a local decision. Similarly, due to the architectural nature of MD-LSTM, the neighboring context around each pixel is taken into account. Furthermore, LSTM itself carries long range contextual information which enables the networks to perform accurate segmentation without any extra processing. In other words, LSTM networks take into account the local (pixel-by-pixel) and global (label-by-label) dependencies in a single process which is a huge advantage for segmentation. Therefore, it skips any additional processing like graphical modeling or a multi-scale pyramid which are commonly used for other segmentation approaches.

The basic architecture consists of an input layer, a hidden layer with LSTM, and an output layer. Additionally, a fully-connected layer can be inserted after the LSTM layer. This layer controls the number of weights and information passing to the next layer when the networks are deeper [Gra12]. Especially when applied on images, this layer acts as a feature mapping layer. More details can be found in Section 6.2.1.

The architecture in this chapter is applied to texture and scene images. First to evaluate the networks on *texture images*, a database of automatically generated blob mixtures of textures is introduced. It generates randomly shaped blobs filled with textures with illumination changes and various transformations in scale, rotation,

and translation. A standard single 2D-LSTM network model is directly applied on the dataset and shows the performance gain compared to other texture segmentation approaches.

Secondly, a deeper and more complex architecture is explored for more challenging set of images, i.e., *natural scene images* containing background clutter and noise. The performances compared to state-of-the-art methods including DL approaches show that the LSTM-based approach achieves better segmentation results with a much lower computational complexity.

The work presented in this chapter appeared in ICIP 2014 [BB14] and CVPR 2015 [BBRL15]. Section 6.1 describes the details of the proposed approach for texture segmentation and its experiments. In section 6.2, a deeper model is presented and evaluated on natural scene images.

6.1 Texture Segmentation

This section presents a network model producing reliable segmentation results compared to other texture segmentation methods. The new dataset is challenging even with human eyes. This section shows how the LSTM-based approach can accurately segment and integrate relative regions without extra processing.

6.1.1 The Approach

The network is designed with an input layer, a single and shallow hidden layer with four LSTM memory blocks. Each pixel passes through these LSTM memory blocks independently before being combined into one activation. After a softmax layer, the networks output a class probability vector of the input pixel. Finally, a pixel class label is predicted based on the maximum score of the output. The size of the input vector (i.e., the number of pixels) closely affects the number of hidden units. The optimal parameters for training in our task will be discussed in Section 6.1.3.

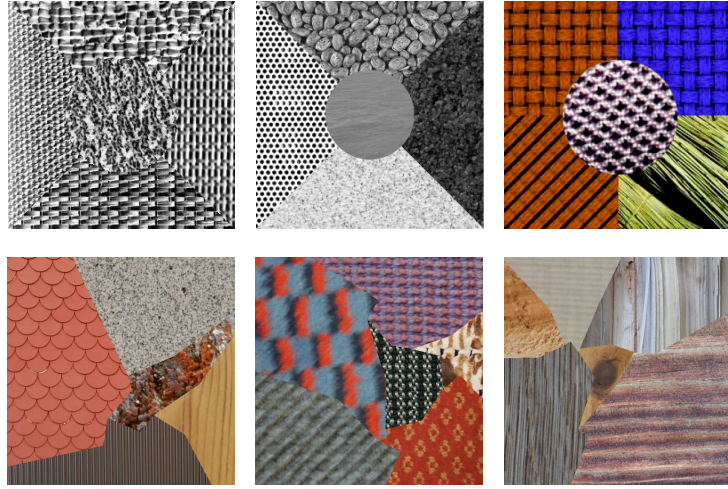


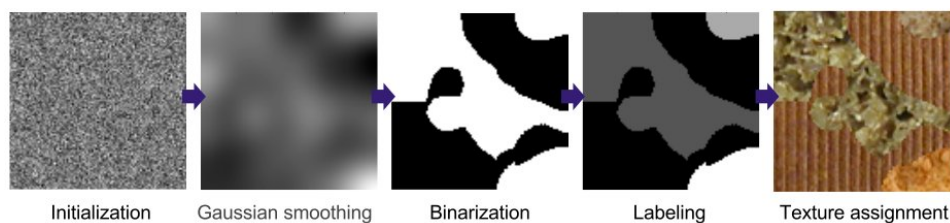
Figure 6.1: Existing texture segmentation datasets. In the first row, the first two images are five texture mosaics from Brodatz album [Bro66], the third image is another five texture mosaics from MIT VisTex dataset [visb], and images in the second row are from the Prague texture segmentation dataset [HM08].

6.1.2 Datasets

(Proposed) Blob-Mosaics Texture Segmentation Database

There are a few public databases for texture segmentation [HM08, USC]. The existing texture segmentation databases are commonly generated by synthetic compositions, which are obtained from a collection of polygon masks. These masks consist of constant Voronoi polygons and is used for all images. This generation technique has some limitations. Since the shape of the regions is static, classifiers tend to learn the shape instead of the actual texture signature. Moreover, the boundary of each texture region includes strong edges and/or corners that affect the performance of segmentation. Therefore, The performance from such datasets does not correspond to the purpose of texture segmentation. Figure 6.1 shows examples of existing texture segmentation datasets facing these issues.

To avoid these issues, a new database using 2D Gaussian blobs is proposed. Images are composed of random 2D Gaussian blobs and each blob is filled with random material textures. To generate blob-mosaic images, a 100×100 sized plain image is first initialized with normal distributed values $[0, 1]$. The initialized noise image is then smoothed by a Gaussian filter ($\sigma = 10.0$). After the binalization with the median value, randomly shaped blobs are generated. Each blob regions are labeled



(a) Blob-mosaics image generation



Figure 6.2: Blob-Mosaics texture segmentation database. The procedure of creating blob-mosaics image is as follows: 1) Gaussian filtering is applied on a randomly initialized image (100×100 pixels with normal distributed values [0-1]). 2) Thresholding is performed for binarization (Median is selected as a threshold value). Randomly shaped regions are generated in this step. 3) Texture images from the KTH-TIPS2-a dataset [KTH] are randomly assigned to the regions. Note that, the dataset KTH-TIPS2-a itself includes material textures with various conditions (different scales, illumination directions, and poses). Thus, the final blob-mosaics images include random shapes in different positions, as well as textures under various transformations and conditions.

and assigned to a random texture. The texture images used here are from KTH-TIPS2-a dataset [KTH], consists of texture images from eleven distinct materials under varying illumination, poses, and scales. Thus, final blob-mosaics images include randomly shaped regions, as well as textures under various conditions, so it provides diverse challenges for texture segmentation. Figure 6.2–a illustrates the flowchart for generating blob-mosaics images along with some example images from the dataset itself. 4000 images are generated for training and the segmentation models are tested on 630 images.

6.1.3 Experimental Setup

All the experiments of LSTM networks have been run by using RNNLIB [Gra]. In order to validate the proposed model, several approaches commonly used for segmentation are compared with the LSTM model.

The baseline methods are listed below:

- Gray-Haralick+Naive Bayes: 6 patch-wise Haralick features (gray) and Naive Bayesian classifier [AK11]. The six Haralick features used here are contrast, energy, homogeneity, correlation, dissimilarity, and angular second moment in four directions, 0° , 45° , 90° , and 135° .
- Color-Haralick+Naive Bayes: patch-wise 13 Haralick combined with color chroma features on the three HSV image channels and Naive Bayesian classifier [AK11, BHSF11a]. The 13 Haralick features used here are angular second moment, contrast, correlation, sum of squares (variance), inverse difference moment, sum average, sum variance, sum entropy, entropy, difference variance, difference entropy, and two information measures of correlation. Note that the maximal correlation coefficient is not included due to some computational instability.
- GMM-HMRF: the combination of a Gaussian Mixture Model(GMM), Expectation Maximization (EM), and Hidden Markov Random Fields (HMRF) [Wan12].

The first comparison method, 6 Haralick features have been extracted on 9×9 patches and each pixel is classified by a Naive Bayesian classifier. The second one consisted on 13 Haralick features with color (color-chroma) in HSV space. For the last comparison, GMM-HMRF, the number of regions ($K = 3, 5$) was initialized, and HMRF-EM was performed on RGB images with 20 EM iterations and 20 MAP iterations. For LSTM networks, several sizes of hidden units ($h = 10, 30, 50, 80$) have been tested. The input and output sizes are set to 3 (Red, green, and blue pixels) and 11 (the number of texture class), respectively. The learning rate of $1e - 5$ and a momentum of 0.9 are fixed for all of our experiments.

Segmentation quality measurement: The pixel-based classification without any spatial information can result in imprecise or noisy segmented area. To measure and judge the robustness of these factors for the proposed methods, segmentation accuracy was measured by area-based quality. Though the area-based accuracy is simply measured by the ratio between the predicted area and the area of corresponding GT; there is no direct way to map the predicted region onto the GT. To find the best possible overlap between them, we first sort the predicted regions from large to small. It helps to avoid a double assignment of regions. The maximum overlapped region between the predicted image and the GT image are matched and the overlapping ratio is the area-quality of the segmentation. Hence, the most probable similarities between the GT image and the predicted image are found and the accuracy of the segmentation per image (Acc_I) is computed as follow:

$$Acc_I = \frac{1}{R} \sum_{r=1}^R \frac{A_r^g \cap A_r^s}{A_r^g},$$

where R is the number of region in the GT, A_r^g is the area with label r in the GT, A_r^s is the maximum portion of the corresponding area for label r in the predicted image, and $A_r^g \cap A_r^s$ is the area of overlapping portion between A_r^g and A_r^s .

6.1.4 Results and Analysis

The best area-based segmentation quality (averaged over the 630 test samples) is compared in Table 6.1. LSTM networks led to superior performance with the best $K = 3$ (for GMM-HMRF) and $h = 30$ (for LSTM networks). Performances with hidden size 30 and 50 are comparable (the difference was only about 0.4%). Segmentation results in Figure 6.8 show the effectiveness of our method. Particularly, many of other approaches have failed except LSTM networks under the difficult blob-mosaics images.

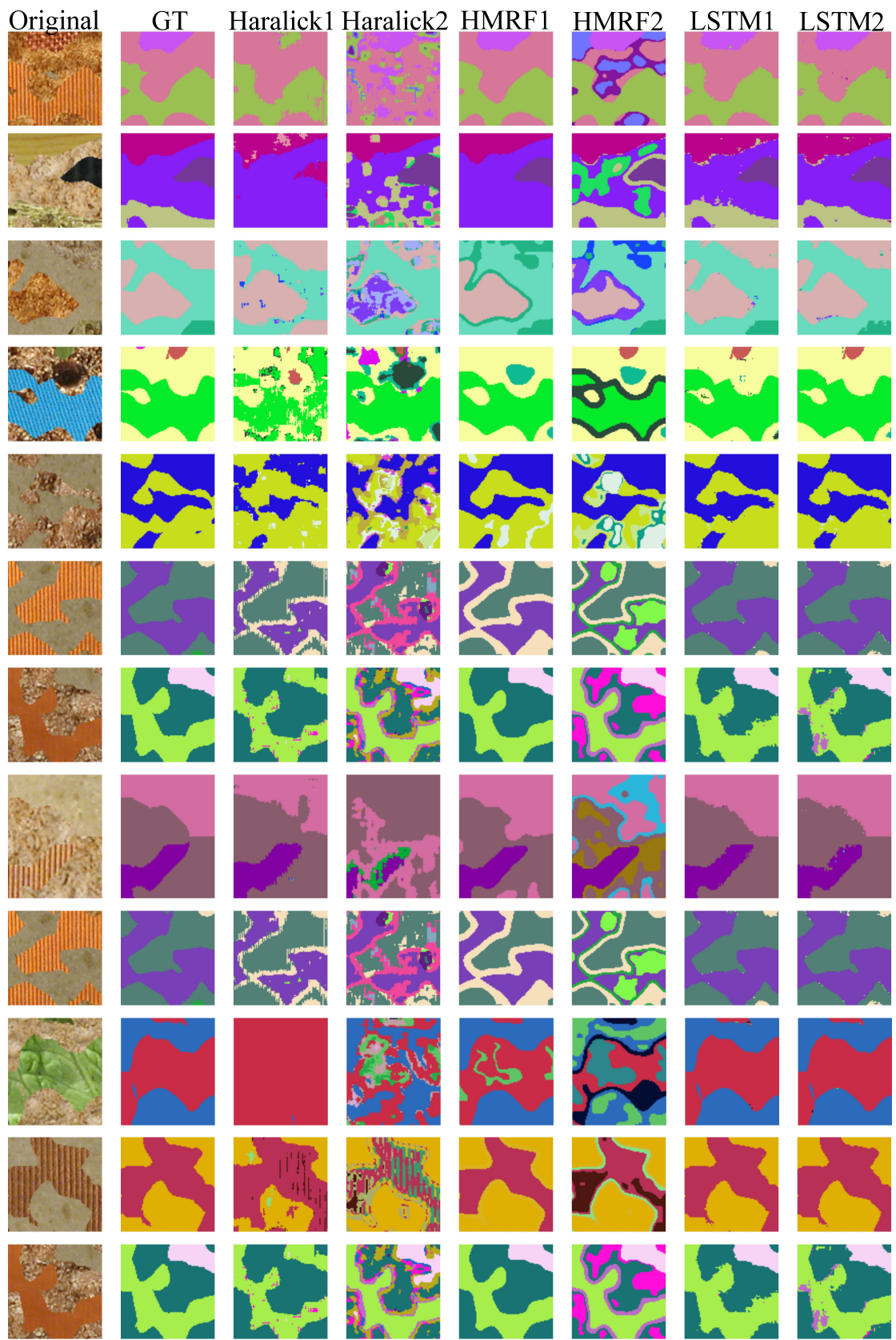
6.1.5 Summary

This section presented a simple way of resolving texture segmentation problem with small LSTM-based networks, i.e., a single hidden layer with 30 hidden units. The task has become challenging by introducing a new texture segmentation dataset

Table 6.1: Accuracy comparison of texture segmentation on texture blob-mosaics images. To compare the segmentation performance, three different methods are selected: (1) patch-wise classification with gray texture features (Gray-Haralick+Naive Bayes), (2) patch-wise classification with gray and color texture features (Color(HSV)-Haralick+Naive Bayes), and (3) Gaussian mixture model+Expectationmaximization+Hidden Markov Random Field (GMM-HMRF). Haralick features are one of the most common texture features extracted from Grey level co-occurrence matrix (GLCM). For GMM-HMR, the best result on the table is with the initial region $K=3$). The accuracy is measured by are-based quality. The details of segmentation quality measurement is explained in Section 6.1.3. The LSTM has obtained the highest average accuracy for texture segmentation. The best score is shown in bold.

method	avg. acc.(%)
Gray-Haralick+Naive Bayes [AK11]	43.87
Color(HSV)-Haralick+Naive Bayes [AK11, BHSF11a]	49.34
GMM-HMRF [Wan12]	71.20
LSTM networks	90.88

where images are random blob-mosaics filled with transformed textures: Scaling, rotation, translation, and illumination are considered. The proposed evaluation criteria is adequate for the blob-mosaics segmentation quality measurement. Finally, our LSTM-based approach outperforms other texture segmentation algorithms using this criteria. In the next section, a deeper network model is designed to cope with high-resolution and noisy input images, naming natural scene images. The system skips any specific pre- or post-processing unlike other DL-based algorithms for such a task.



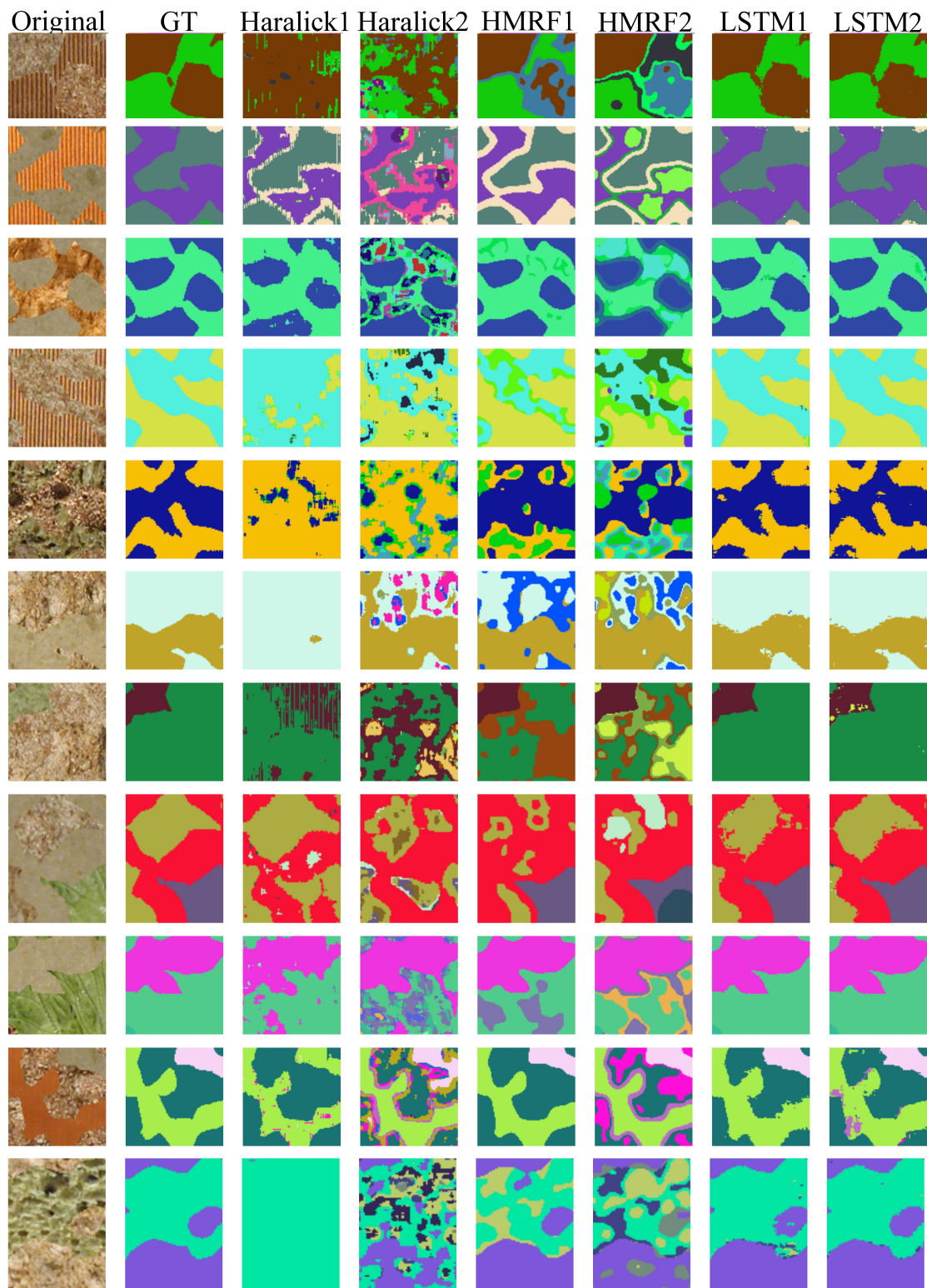


Figure 6.3: Segmentation results of blob-mosaics images. From left to right column are original image (Original), Ground-Truth (GT), Haralick gray features with Naive Bayesian classifier (Haralick1), Haralick color features with Naive Bayesian classifier (Haralick2), and HMM-HMRF (The initial region $K = 3$ (HMRF1) and 5 (HMRF2)), and LSTM networks (learning rate (lr) = $1e-5$, hidden size (h) = 30 (LSTM1) and 50 (LSTM2)). The segmentation results show the superior performance of the proposed method.

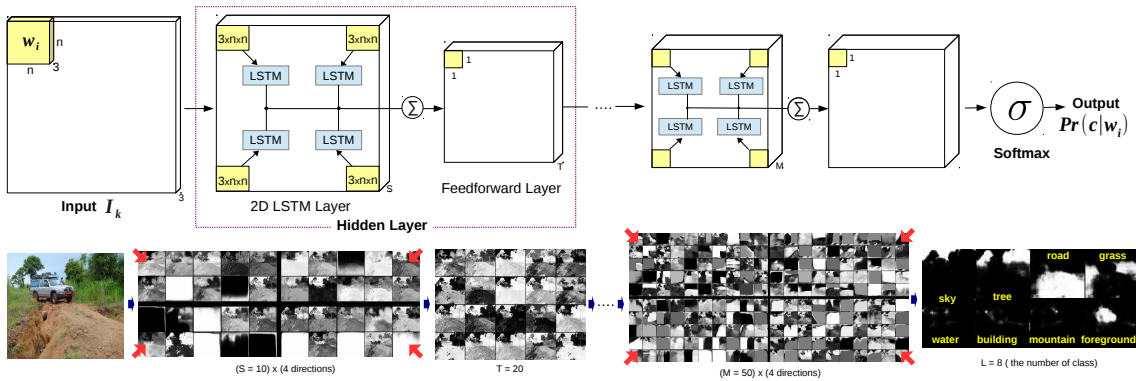


Figure 6.4: 2D-LSTM network architecture. An input image I_k is divided into non-overlapping windows w_i (a grid sized $n \times n$). Each window per RGB channels ($3 \times n \times n$) is fed into four separate LSTM memory blocks. The current window of an LSTM block is connected to its surrounding directions x and y , i.e., left-top, left-bottom, right-top, and right-bottom; it propagates surrounding contexts. The output of each LSTM block is then passed on the feed-forward layer, which sums all directions and squash it by the Hyperbolic tangent (\tanh). At the last layer, the outputs of the final LSTM blocks are summed up and sent to the *softmax* layer. Finally, the networks output the class probabilities ($Pr(c|w_i)$) for each input window. The bottom images are corresponding outputs for each layer.

6.2 Scene Labeling

Image segmentation consists in grouping the relevant pixels into a region and assign a corresponding label for that region. Scene labeling is a segmentation task especially for natural scene images, which are in general very diverse in resolution, quality, and contents. As mentioned in Chapter 3, LSTM networks are limited to learn long-term dependencies which can most likely occur in such data. This section presents how a LSTM handles this issue and learns instances with large variation within a class. 2D-LSTM takes into account local (pixel-by-pixel) and global (label-by-label) contextual information in a single process. In other words, it can skip any additional processing or conditions like multi-scale or different patch sizes to solve the scene labeling task with minimum human and machine effort. The experiments show how LSTM networks generalized well for any vision-based task and efficiently learn without any task-specific features.

6.2.1 The Approach

The input is first divided into a non-overlapping grid of size $n \times n$, ($n > 1$). The values in an image tend to have gradual changes (within 2-3 pixels). The input with small windows keep the local context and reduce the burden of learning long-term dependencies for the network. More detailed explanation of this phenomenon is in Section 4.1.1. For one window input w_i , four LSTM modules carry the information from each direction (left-top, right-top, left-down, and right-down). These are added and sent to the fully connected (Feed forward) layer. These steps (the LSTMs and the fully connected layer) can be repeated many times. At the end, the activation passes the softmax layer and get the final class probabilities of the window, w_i : $Pr(c|w_i)$. A standard loss function like cross-entropy error function (Equation (4.2)) computes the error between a predicted probability vector and a true probability vector (1 for the true label and 0 for others, 1-to-K target coding). Since the size of the input is more than one pixel and each pixel has a corresponding label ($n \times n$ labels from $n \times n$ input pixels), the true probability vector can also be represented as probabilities of the occurrence of classes within the window w_i . For that reason, probabilistic target coding is applied for the output layer. Section 4.1.3 explains in more detail the coding scheme. Figure 6.4 visualizes the process explained above, and Figure 6.6 shows the behavior of learning process.

One important factor with this architecture is that the fully connected layer acts as a feature mapping from a 2D-LSTM layer, so the amount of features from all contextual information on the image is generated and combined together in this layer. The size of the layer corresponds to the number of feature maps; the bigger the size of feed-forward layer the more features it creates. Figure 6.5 shows what types of contextual information are learned in each layer. More detailed features are created in lower levels and more abstract and complex features are focused at higher levels with global contexts.

6.2.2 Datasets

Our approach has been tested on two fully labeled outdoor scene datasets: the Stanford Background dataset [GFK09a] and the SIFT Flow dataset [LYT11]. The Stanford Background dataset contains 715 images composed of 8 common labels chosen from existing public datasets (572 images used for training, the rest for testing).

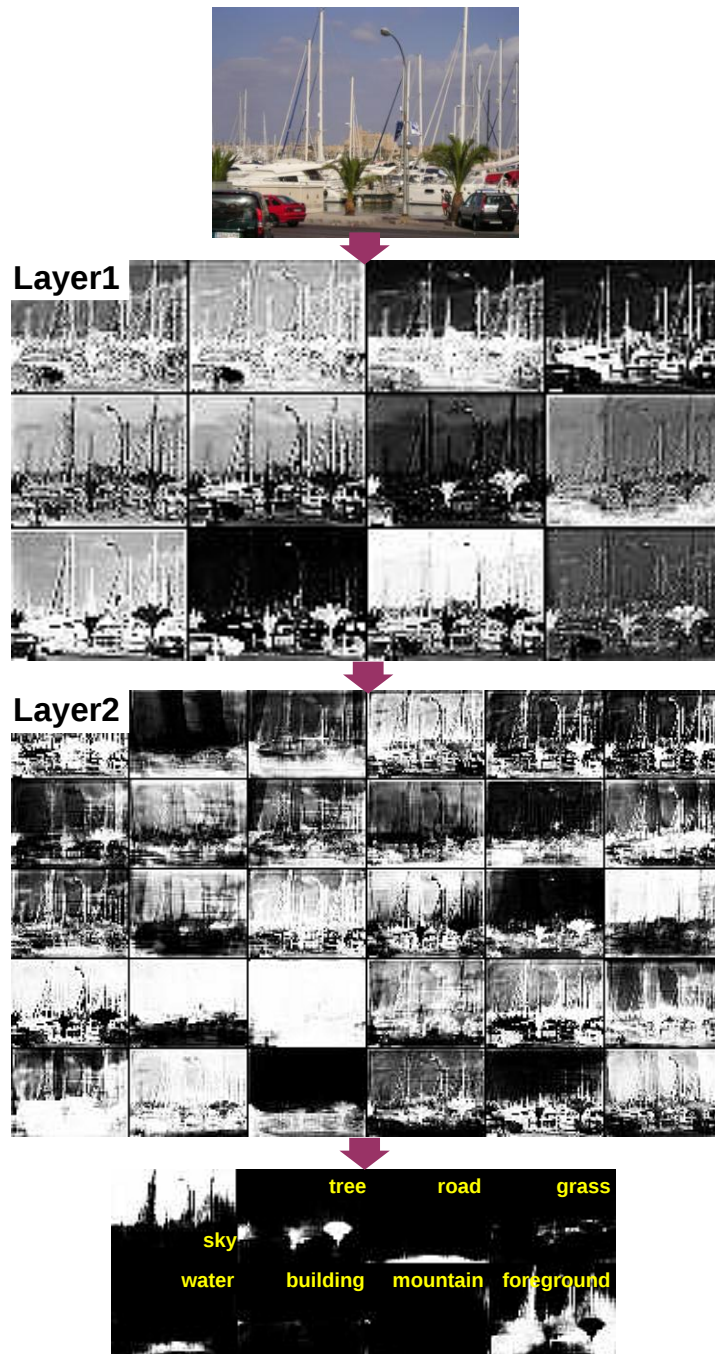


Figure 6.5: Visualization of feature maps in each layer. The activations are sampled after the convolutional summing of four LSTM blocks. Each activation from the input window is projected down to the image space. Each image represents the features from each hidden node of the corresponding layer. Note that a lighter color represents higher activations.

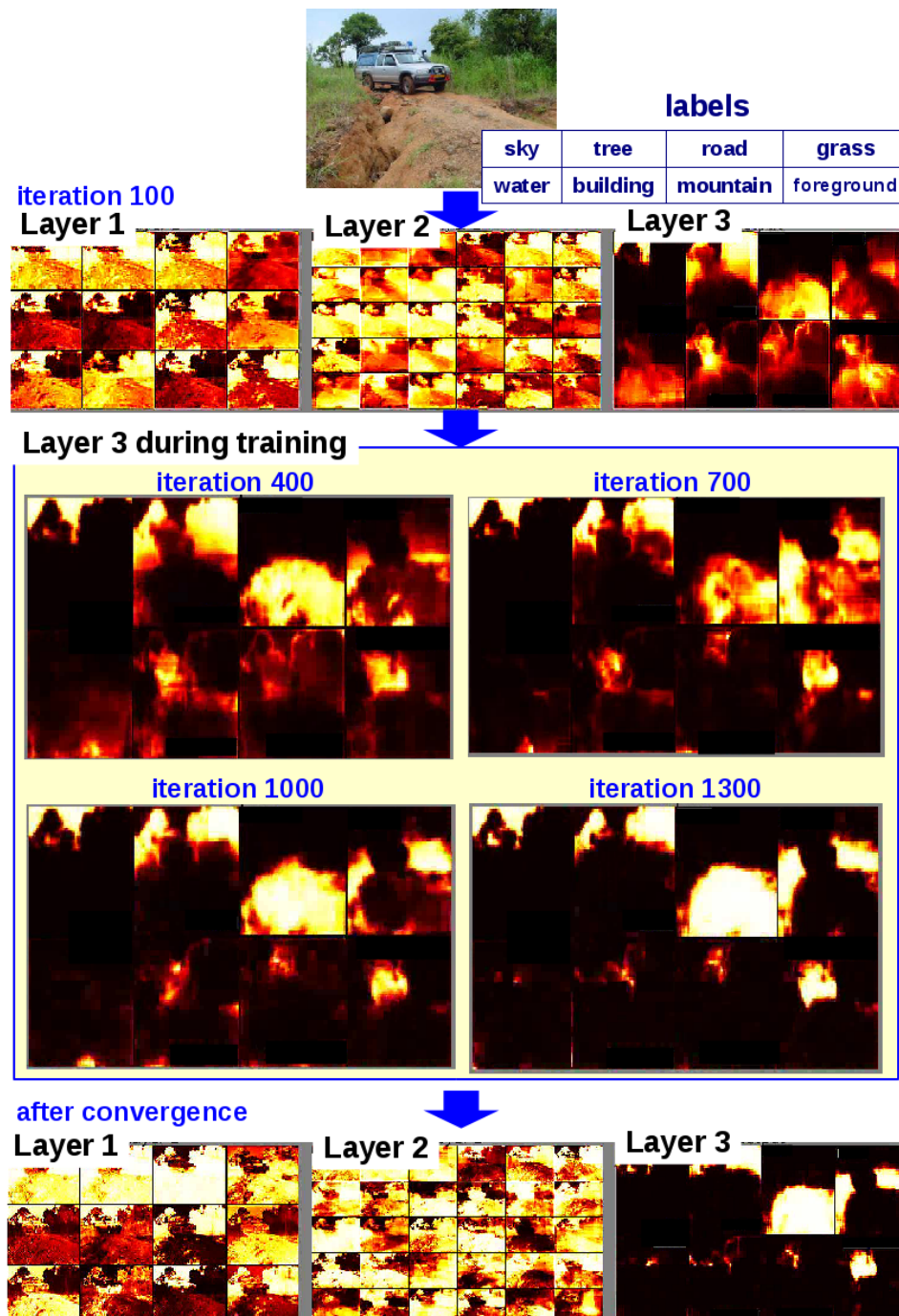


Figure 6.6: The behavior changes of output activations from the networks while training for scene labeling, and after convergence. The first feature map (at iteration 100) and the last feature map (after convergence) are also compared. All activations are captured every 300 iterations. Lighter colors correspond to higher activations.

Note that one of the labels, “foreground” contains unknown objects. Each image has a different resolution (on average 320×240). The SIFT Flow dataset consists of 2688 images of 256×256 pixels each. The dataset is split into 2488 images used for training and the rest for testing. The images include 33 semantic labels labeled by LabelMe users.

6.2.3 Experimental Setup

All LSTM network models in our experiments have three layers. The second and third layer have 20 LSTM units with 30 units in the feed-forward layer and 50 LSTM (no feed-forward layer at the final level) respectively. In the first layer, the hidden size was decided based on the size of window, i.e., 4 LSTM with 10 units in the feed-forward layer for 3×3 input windows and 10 LSTM with 12 units in the feed-forward layer for 5×5 input windows. At the end, the output of the networks is reduced by a factor of window-size, which is up-scaled with cubic interpolation for the final evaluation. On-line gradient descent was used for training with a learning rate of 10^{-6} and a momentum of 0.9.

Evaluation: Both pixel accuracy and class-average accuracy are reported to compare our performance with other approaches. The *pixel accuracy* measures the ratio of true positive pixels over all pixels, and the *class-average accuracy* averages over all class accuracies using an equal weight for all classes. The measure of class-average accuracy has more impact when the classes are imbalanced in test images, which is common on outdoor scene images, e.g., most scenes contain “sky” but not “tree”.

6.2.4 Results and Analysis

Table 6.2 compares the performance of LSTM networks with the current state-of-the-art methods on the Stanford Background dataset and the SIFT Flow dataset. Note that the compared methods combined with pre- or post-processing and multi-scaled ones are not considered in the table to make the comparison as fair as possible. However, single-scale LSTM networks are still comparable to the multi-scale versions of state-of-art-methods, e.g., pixel accuracy of 78.8% for multi-scale CNNs [FCNL13], 76.36% for multi-scale augmented CNNs [KEF⁺14], and 78.56% for single-scale

Table 6.2: Pixel and averaged per class accuracy comparison on the Stanford Background dataset (top) and the SIFT Flow dataset (bottom). The best scores under unbalanced class frequency are shown in bold. For the Stanford Background dataset, the approaches which include pre- or post-processing, and/or multi-scale pyramids are not reported here as they cannot be directly compared (for the SIFT Flow dataset, ConvNet only reported accuracy on a multi-scale version [FCNL13]). LSTM networks lead to high performance with a very fast inference time on CPU. Balancing the class frequencies of input images would improve the class-average accuracy, but is not realistic for scene labeling in general. The performance of RCNN reported here is from two instances. For more details, see Section 6.2.4. CT and W indicate the average computing time per image and window size respectively. Baseline methods: Superparsing [TL10], Singlescale and Multi-scale ConvNet [FCNL13], Augmented CNNs [KEF⁺14], and Recurrent Convolutional Neural Networks (RCNN) [PC14]

Stanford Background dataset					
Method	Pixel Acc. (%)	Class Acc. (%)	Class frequency	CT (sec.)	# parameters
Superparsing	77.5	-	-	10 to 300	-
Singlescale ConvNet	66	56.5	balanced	0.35 (GPU)	-
Augmented CNNs	71.97	66.16	balanced	-	701K
Recurrent CNNs	76.2	67.2	unbalanced	1.1 (GPU)	-
LSTM networks (W 5×5)	77.73	68.26	unbalanced	1.3 (CPU)	173K
LSTM networks (W 3×3)	78.56	68.79	unbalanced	3.7 (CPU)	155K
SIFT Flow dataset					
Method	Pixel Acc. (%)	Class Acc. (%)	Class frequency	CT (sec.)	# parameters
Multi-scale ConvNet	67.9	45.9	balanced	-	-
Augmented CNNs	49.39	44.54	balanced	-	1225K
Recurrent CNNs	65.5	20.8	unbalanced	-	-
LSTM networks (W 5×5)	68.74	22.59	unbalanced	1.2 (CPU)	178K
LSTM networks (W 3×3)	70.11	20.90	unbalanced	3.1 (CPU)	168K

LSTM networks. Multi-scale CNNs with further post-processing, i.e., super-pixel, CRF, and segmentation tree, slightly improve the accuracy, but are around 15-100 times slower than without post-processing.

With RCNNs, two instances¹ are considered for the accuracy comparison. Note that higher network instances increase the context patch size to correct a final prediction. An increase in the number of instances maintains the capacity of the system constant (since the network weights are shared), but it causes a dramatic growth in training time. The testing time is reported as increasing tenfold when one more instance added to the network. With pixel accuracy, LSTM networks perform about 2 percentage points better compared to RCNN with two instances, but around 2 percentage points worse than RCNN with three instances on the Stanford Background dataset. On the SIFT Flow dataset, the accuracy of LSTM networks is around 4 percentage points higher than RCNN with two instances, but around 7 percentage points lower than with three instances. However, the differences of average per class accuracy are less than 1% with both two and three instances on all datasets. Overall, our LSTM network model is efficient in training and testing — LSTM networks do not need time-consuming computations to combine long-range context information. LSTM achieves results higher than state-of-the-art methods without any extra effort in an end-to-end manner.

The confusion matrix on the Stanford Background dataset is reported in Figure 6.7. “mountain” is the hardest class (only 9.5% class accuracy), but this is explained by the small size of the training and the testing sets compared to other classes (see the class frequency distribution in Figure 6.7). Note that other methods solved this issue by balancing class frequencies, yet doing so is not realistic. The labels being confused most often are “mountain” with “tree” or “building”, and “water” with “road”. It is clear from the class frequency distributions in Figure 6.7 that “mountain” and “water” have the least frequent samples in both the training and testing sets. The visual labeling result of confused classes is shown in Figure 6.9. These are mostly well-segmented but mislabeled.

Selected examples of labeling results from the Stanford dataset are shown in Figure 6.8. Our approach yields very precise pixel-wise labeling. Figure 6.9 shows an example of misclassification with the corresponding GT. The results in the first and

¹RCNNs compute a sequential series of networks. The networks train various sizes of the same input image recurrently in order to learn increasingly large contexts for each pixel. For more details, see the original paper [PC14].

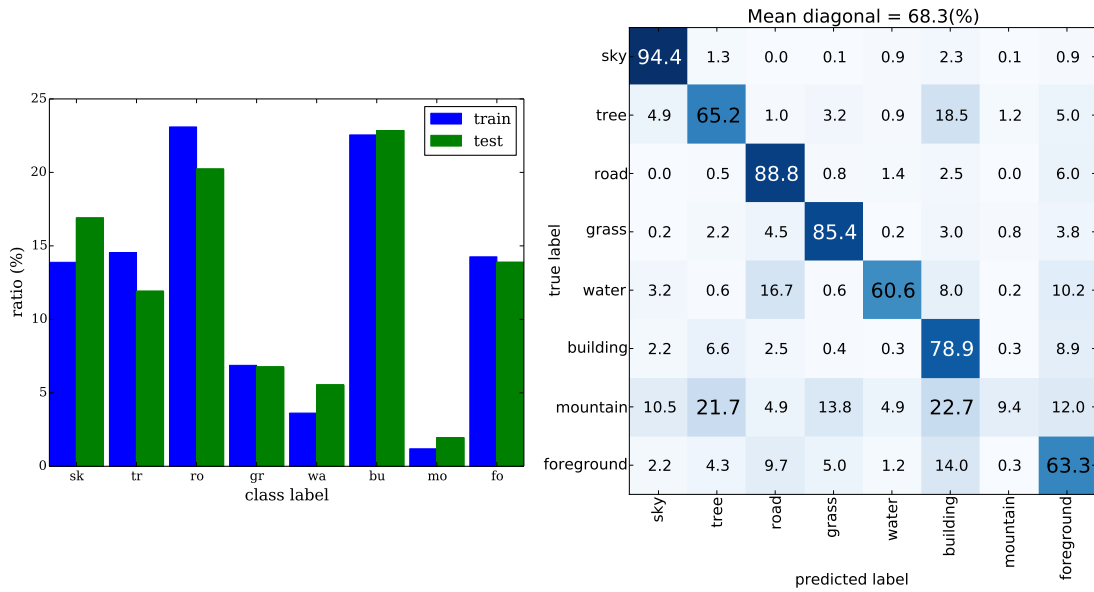


Figure 6.7: The left figure shows the distribution of class frequencies of the Stanford Background dataset (the ratio of all classes sums to 100%). Categories of the dataset are sky (sk), tree (tr), road (ro), grass (gr), water (wa), building (bu), mountain (mo), and background. As it can be seen, there are similar distributions between training and testing data, but diverse between classes. For instance, the difference between the highest distribution “building” and the lowest distribution “mountain” is of about 20 percentage points. The figure on the right shows the confusion table on the Stanford Background dataset. As it can be observed, the percentage of misclassification labels is correlated with the frequency of the class in the dataset. For instance, the training and testing sets contain only 2% of “mountain” labels and less than 5% of “water” and “grass” labels. As a consequence, “mountain” was confused with “tree” and “building”, “water” with “road”, and “tree” with “building”. From these mislabeling results, we can observe that the wrong predictions are often caused in the label prediction level but not the segmentation level (some examples are shown in Figure 6.9).

second row are mislabeled in the GT image (human mistakes), but correctly classified by LSTM networks. The misclassification regions from the third row mostly include very foggy mountains, so it is not visible even to the human eye. The results from the fourth to sixth rows show a precise segmentation but an incorrect labeling (because of the ambiguity of the label’s characteristics). All of these examples reflect challenges of the datasets and the task itself.

6.2.5 Summary

This section shows how deep LSTM networks excel on the task of where large images dealing with the long-term dependencies. Each pixel on the complex and noisy scene images is accurately labeled by combining local and global context within a network model. The experiments show that LSTM networks outperform other state-of-the-art approaches including DL-based algorithms. Furthermore, only with a CPU implementation, the running time of our approach is comparable or better than other state-of-the-art approaches that use GPU.

6.3 Conclusion

This chapter presented an entirely learning-based approach for image segmentation. The architecture for segmentation is simple and well-adapted on different type of images: texture and scene images. Both a single-layer and a deeper network has been successfully applied for texture and scene images, and show performance gains without any hand-crafted features, pre- or post-processing techniques, and multi-scale pyramids from input images. Beside, this architecture has a (comparatively) fast training and testing time on a single-core CPU, as it uses a smaller number of parameters than other DL approaches.

In the next chapter, the architecture for segmentation with an advanced LSTM model (PyraMiD-LSTM) for 3D volumetric data will be introduced. PyraMiD-LSTM is easy to parallelize, especially for 3D data. Hence, fast biomedical volumetric images segmentation with PyraMiD-LSTM will be presented.

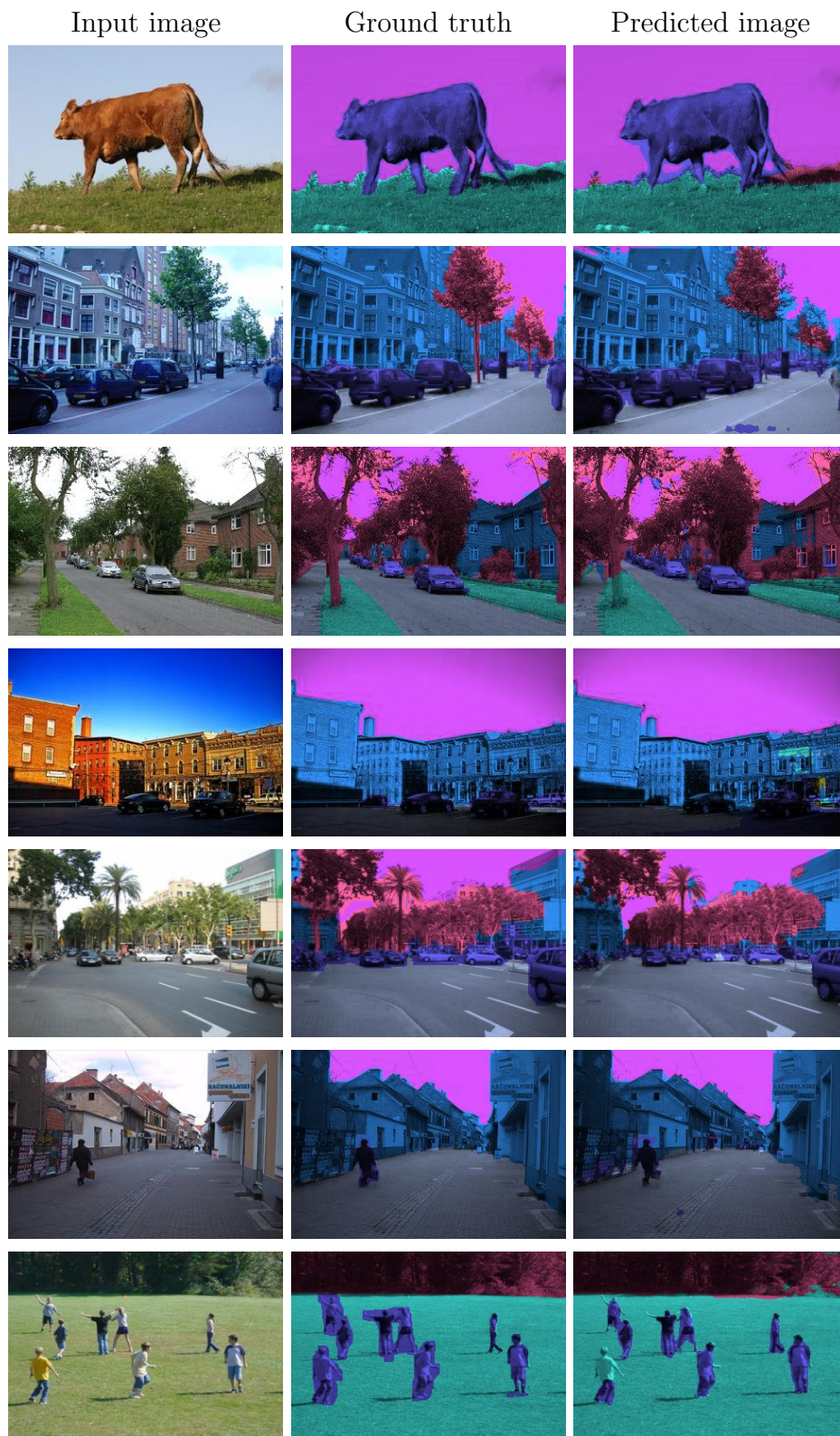


Figure 6.8: The results of scene labeling on the Stanford Background dataset. First column: input image; Second column: GT; Third column: predicted image. Colors on images indicate labels, and the predicted images indicate a correct labeling. There are identical colors on the GT images and the predicted images.



Figure 6.9: Selected mislabeled examples from LSTM networks. The first and second errors are mainly from mislabeling in the GT: human mistake — the car and human parts are labeled as “road” instead of its label “foreground” in the GT. The third misclassified regions are from very foggy forest. Furthermore, the reflection of a car wheel on the water is misclassified as a wheel (label “foreground”). These examples are understandable mistakes. The results show the difficulties of our dataset and the ambiguity of actual labels. The fourth to sixth examples show common mistakes of LSTM networks — the well-segmented regions are mislabeled. First row: road (gt) to grass (predicted); Second row: water (gt) to road (predicted); Third row: grass (gt) to road (predicted)

Chapter 7

Parallel Volumetric LSTM Networks

In this chapter, PyraMiD-LSTM proposed in Chapter 3.3 is applied to biomedical volumetric image segmentation. As mentioned earlier, a great benefit of LSTM is to take into account the local and global context of each pixel through all pixels. The last two chapters have presented MD-LSTM for 2D image. They show that LSTM brings strong correlation along the x and y axes and can effectively learn two-dimensional context. Therefore, MD-LSTM applicability can theoretically be extended to 3D or 4D data. However, the computational cost of processing the entire context in LSTM is a serious drawback. Moreover, the natural architecture of LSTM involves a recurrent connection which processes the meaningful interdependencies of the continuous. Due to this characteristic, it is hard to parallelize on GPUs, unlike CNNs. PyraMiD-LSTM instead introduces the independencies of its computational flow by changing its connection topology, which makes the parallelization feasible.

This work presented in this chapter appeared in NIPS 2015 [SBLS15] and Marijn F. Stollenga was equally contributed to this work. The following section presents the details of the approach and the experiments on 3D biomedical data.

7.1 Biomedical Volumetric Image Segmentation

Though biomedical image analysis is an important topic in the field of biology and medical science, there are limited approaches to achieve fully-automatic and accurate segmentation. Biomedical images, such as MR brain images and EM images, contain a large amount of noise and are mostly volumetric (see Figure 1.6). As mentioned in Section 1.1.1, most previous approaches treat these volumetric data as 2D slices and apply some image segmentation algorithms commonly used in CV field.

Three Dimensional LSTM (3D-LSTM), however, can process the full context of each pixel in such a volume through 8 sweeps over all pixels by 8 different LSTMs, each sweep in the general direction of one of the 8 directed volume diagonals. Here, the novel PyraMiD-LSTM, a variant of 3D-LSTM, uses a rather different topology and update strategy which is easier to parallelize, needs fewer computations overall, and scales well on GPU architectures.

7.1.1 The Approach

The network contains an input layer, several hidden layers including C-LSTM layers and fully connected layers, and an output layer with softmax. The network architecture for 3D volumetric image segmentation is shown in Figure 4.2-d. Randomly rotated and flipped input is sampled from random locations before fed to C-LSTM. Note that the data augmentation is performed during training, but not testing. C-LSTM performs a LSTM computations with convolution operation in a plain. Six C-LSTM compute the LSTM gates and the memory cell over three axes (x, y, z) : $\{(\cdot, \cdot, 1), (\cdot, \cdot, -1), (\cdot, 1, \cdot), (\cdot, -1, \cdot), (1, \cdot, \cdot), (-1, \cdot, \cdot)\}$. The standard MD-LSTM, whose operations are matrix multiplication, has the recurrent connections along with the axes. Unlike the standard MD-LSTM, C-LSTM use convolution to perform LSTM operations, and the recurrent connections are in a plane. Figure 3.4 shows the context information flow of MD-LSTM and PyraMiD-LSTM. As can be seen in Figure 3.5, the connection topology in MD-LSTM becomes a cube and in PyraMiD-LSTM becomes a pyramid. The outputs from all C-LSTMs are combined and sent to the fully connected layer. \tanh is used as a squashing function in the hidden layer. Several of these C-LSTM and fully connected layers, i.e., PyraMiD-LSTM layers, are stacked before an output layer. The last layer is fully-

connected and uses a softmax function to compute probabilities for each class for each pixel. See Section 3.3 for the details of the computations.

7.1.2 Datasets

The approach is evaluated on two 3D biomedical image segmentation datasets: EM and MR Brain images.

Electron Microscopy (EM) dataset: The EM dataset [CSP⁺10] is provided by the ISBI 2012 workshop on Segmentation of Neuronal Structures in EM Stacks [Seg12]. Two stacks consist of 30 slices of 512×512 pixels obtained from a $2 \times 2 \times 1.5 \mu\text{m}^3$ microcube with a resolution of $4 \times 4 \times 50 \text{ nm}^3/\text{pixel}$ and binary labels. One stack is used for training, the other for testing. Target data consists of binary labels (membrane and non-membrane).

Magnetic Resonance (MR) Brain dataset: The MR Brain images are provided by the ISBI 2015 workshop on Neonatal and Adult MR Brain Image Segmentation (ISBI NEATBrainS15) [A. 15]. The dataset consists of twenty fully annotated high-field (3T) multi-sequences: 3D T1-weighted scan (T1), T1-weighted inversion recovery scan (IR), and fluid-attenuated inversion recovery scan (FLAIR). The dataset is divided into a training set with five volumes and a test set with fifteen volumes. All scans are bias-corrected and aligned. Each volume includes 48 slices with 240×240 pixels (3mm slice thickness). The slices are manually segmented through nine labels: cortical gray matter, basal ganglia, white matter, white matter lesions, cerebrospinal fluid in the extracerebral space, ventricles, cerebellum, brainstem, and background. Following the ISBI NEATBrainS15 workshop procedure, all labels are grouped into four classes and background: 1) cortical gray matter and basal ganglia (GM), 2) white matter and white matter lesions (WM), 3) cerebrospinal fluid and ventricles (CSF), and 4) cerebellum and brainstem. Class 4) is ignored for the final evaluation as required.

7.1.3 Experimental Setup

All experiments are performed on a desktop computer with an NVIDIA GTX TITAN X 12GB GPU. For GPU implementation, the NVIDIA CUDA Deep Neural Network library (cuDNN) [CWV⁺14] is used. On the MR brain dataset, training took around three days, and testing per volume took around 2 minutes.

Exactly the same hyper-parameters and architecture are used for both datasets. Our networks contain three PyraMiD-LSTM layers. The first PyraMiD-LSTM layer has 16 hidden units followed by a fully-connected layer with 25 hidden units. In the next PyraMiD-LSTM layer, 32 hidden units are connected to a fully-connected layer with 45 hidden units. In the last PyraMiD-LSTM layer, 64 hidden units are connected to the fully-connected output layer whose size equals the number of classes.

The convolutional filter size for all PyraMiD-LSTM layers is set to 7×7 . The total number of weights is 10,751,549, and all weights are initialized according to a uniform distribution: $\mathcal{U}(-0.1, 0.1)$.

Training: RMS-prop with momentum [TH12] is applied here (see Section 4.3.5).

The error function is the squared loss:

$$E = (y^* - y)^2, \quad (7.1)$$

where y^* is the target, y is the predicted output from the network. The squared loss is chosen here as it produced better results than using other error function like the log-likelihood error function 4.2.

A decaying learning rate is used: $\lambda_{lr} = 10^{-6} + 10^{-2} \left(\sqrt[100]{\frac{1}{2}} \right)^{epoch}$, which starts at $\lambda_{lr} \approx 10^{-2}$ and halves every 100 epochs asymptotically towards $\lambda_{lr} = 10^{-6}$. Other hyper-parameters used are $\epsilon = 10^{-5}$, $\rho_{MSE} = 0.9$, and $\rho_M = 0.9$.

Sub-volumes and augmentation: The full dataset requires more than the 12 GB of memory provided by our GPU, hence training and testing are performed on sub-volumes. A position in the full data and extract a smaller cube are randomly picked (see the details in *Bootstrapping*). This cube is possibly rotated at a random angle over some axis and can be flipped over any axis. EM images are rotated over

the z-axis and flipped sub-volumes with 50% chance along x, y, and z axes. For MR brain images, rotation is disabled; only flipping along the x direction is considered, since brains are (mostly) symmetric in this direction.

During test-time, rotations and flipping are disabled and the results of all sub-volumes are stitched together using a Gaussian kernel, providing the final result.

Pre-processing: Each input slice is normalized towards a mean of zero and variance of one, since the imaging methods sometimes yield large variability in contrast and brightness. The complex pre-processing common in biomedical image segmentation [WGS⁺15] is not applied.

The only prior processing is simple pre-processing on the three datatypes of the MR Brain dataset, since they contain large brightness changes under the same label (even within one slice; see Figure 7.2). From all slices the Gaussian smoothed images are subtracted (filter size: 31×31 , $\sigma = 5.0$), then a Contrast-Limited Adaptive Histogram Equalization (CLAHE) [PAA⁺87] is applied to enhance the local contrast (tile size: 16×16 , contrast limit: 2.0). An example of the images after pre-processing is shown in Figure 7.2. The original and pre-processed images are all used, except the original IR images (Figure 7.2-b), which have high variability.

Bootstrapping: To speed up training, three learning procedures are performed with increasing sub-volume sizes: first, 3000 epochs with size $64 \times 64 \times 8$ and then 2000 epochs with size $128 \times 128 \times 15$. Finally, for the EM-dataset, we train 1000 epochs with size $256 \times 256 \times 20$, and for the MR Brain dataset 1000 epochs with size $240 \times 240 \times 25$. After each epoch, the learning rate λ_{lr} is reset.

Evaluation: For EM images, three error metrics evaluate the following factors:

- Rand error [Ran71]: 1 - F-score of rand index, which measures similarity between two segmentations on the foreground.
- Warping error [JBR⁺10]: topological disagreements (object splits and mergers)
- Pixel error: 1 - F-score of pixel similarity

Table 7.1: Performance comparison on EM images. Some of the competing methods reported in the ISBI 2012 website are not yet published. Comparison details can be found under http://brainiac2.mit.edu/isbi_challenge/leaders-board.

Group	Rand Err.	Warping Err. ($\times 10^{-3}$)	Pixel Err.
Human	0.002	0.0053	0.001
Simple Thresholding	0.450	17.14	0.225
IDSIA (CNNs) [CGGS12]	0.050	0.420	0.061
DIVE	0.048	0.374	0.058
PyraMiD-LSTM	0.047	0.462	0.062
IDSIA-SCI	0.0189	0.617	0.103
DIVE-SCI	0.0178	0.307	0.058

For Magnetic Resonance (MR) brain images, the results are compared based on the following three measures:

- The DICE overlap (DC) [Dic45]: spatial overlap between the segmented volume and ground truth
- The modified Hausdorff distance (MD) [HKR93]: 95th-percentile Hausdorff distance between the segmented volume and ground truth
- The absolute volume difference (AVD) [BPA⁺09]: the absolute difference between segmented and ground truth volume, normalized over the whole volume.

7.1.4 Results and Analysis

Membrane segmentation is evaluated through an on-line system provided by the ISBI 2012 organizers. The measures used are the Rand error, warping error and pixel error [Seg12]. Comparisons to other methods are reported in Table 7.1. The teams IDSIA and DIVE provide membrane probability maps for each pixel, like our method. Note that, the IDSIA team uses a state-of-the-art deep CNNs [CGGS12]. These maps are adapted by the problem-specific post-processing technique of the teams SCI [LJST14], which directly optimizes the rand error (DIVE-SCI (top-1) and IDSIA-SCI (top-2)); this is most important in this particular segmentation task.

Without post-processing, PyraMiD-LSTM networks outperform other methods in rand error, and are competitive in wrapping and pixel errors. Of course, performance could be further improved by applying post-processing techniques. Figure 7.1 shows some examples of segmentation results of PyraMiD-LSTM and the comparison with CNNs [CGGS12]. Both results are without any problem-specific post-processing. However, some mild post-processing is applied for CNNs [CGGS12]. The mild post-processing technique used in [CGGS12] is in the following. First, the output of CNNs are calibrated with a polynomial function. After calibration, the output is spatially smoothed by a median filter. As can be seen in Figure 7.1, the segmentation results from PyraMiD-LSTM are cleaner, smooth, more accurate, as well as have less broken edges without any post-processing compared to CNN.

MR brain image segmentation results are evaluated by the ISBI NEATBrain15 organizers [A. 15] who provided the extensive comparison to other approaches on <http://mrbrains13.isi.uu.nl/results.php>. Table 7.2 compares our results to those of the top five teams. The organizers compute nine measures in total and rank all teams for each of them separately. These ranks are then summed per team, determining the final ranking (ties are broken using the standard deviation). PyraMiD-LSTM leads the final ranking with a new state-of-the-art result and outperforms other methods for Cerebrospinal Fluid in all metrics.

Table 7.2: Performance comparison on MR brain images.

Structure	Gray Matter			White Matter			Cerebrospinal Fluid			Rank
Metric	DC (%)	MD (mm)	AVD (%)	DC (%)	MD (mm)	AVD (%)	DC (%)	MD (mm)	AVD (%)	
BIGR2	84.65	1.88	6.14	88.42	2.36	6.02	78.31	3.19	22.8	6
KSOM GHMF	84.12	1.92	5.44	87.96	2.49	6.59	82.10	2.71	12.8	5
MNAB2	84.50	1.69	7.10	88.04	2.12	7.73	82.30	2.27	8.73	4
ISI-Neonatology	85.77	1.62	6.62	88.66	2.06	6.96	81.08	2.66	9.77	3
UNC-IDEA	84.36	1.62	7.04	88.69	2.06	6.46	82.81	2.35	10.5	2
PyraMiD-LSTM	84.82	1.69	6.77	88.33	2.07	7.05	83.72	2.14	7.10	1

7.2 Conclusion

This chapter introduces a parallelizable variant of MD-LSTM, PyraMiD-LSTM, which resolves a problem that the highly promising architecture of MD-LSTM has; previous MD-LSTM implementations could not exploit the parallelism of modern GPU hardware. This issue has been improved through a novel sequential flow of information proposed here, which arguably makes parallelization much easier. The method is evaluated on two challenging benchmarks for biological volumetric image analysis, and has achieved state-of-the-art segmentation results.

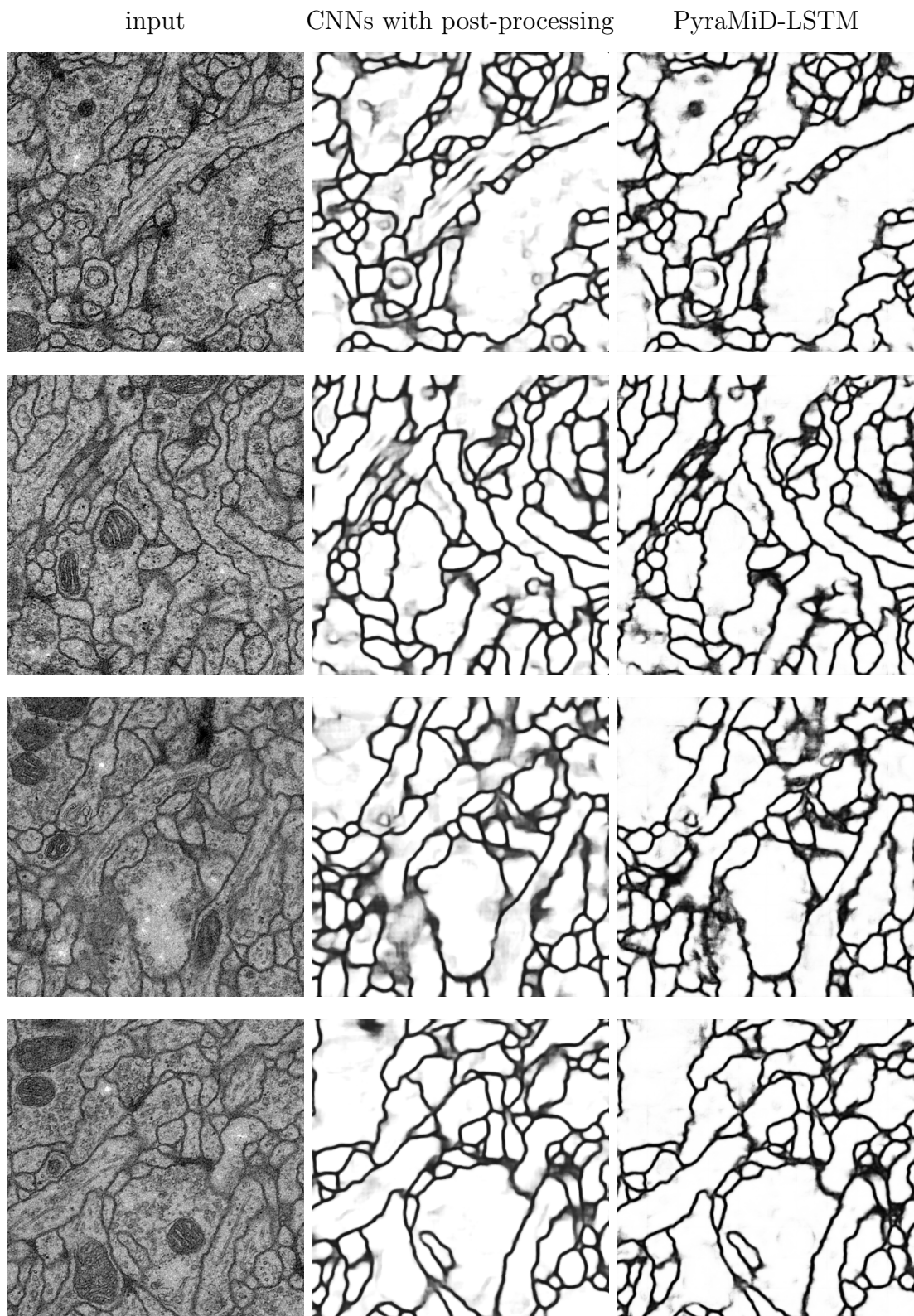


Figure 7.1: Segmentation results for CNNs and PyraMiD-LSTM on EM dataset (slice 8, 13, 24, 27). The results from CNNs are by Ciresan et al. [CGGS12]. In this work, a polynomial function post-processor is applied to the CNN' outputs. The results from PyraMiD-LSTM is the direct output from the network.

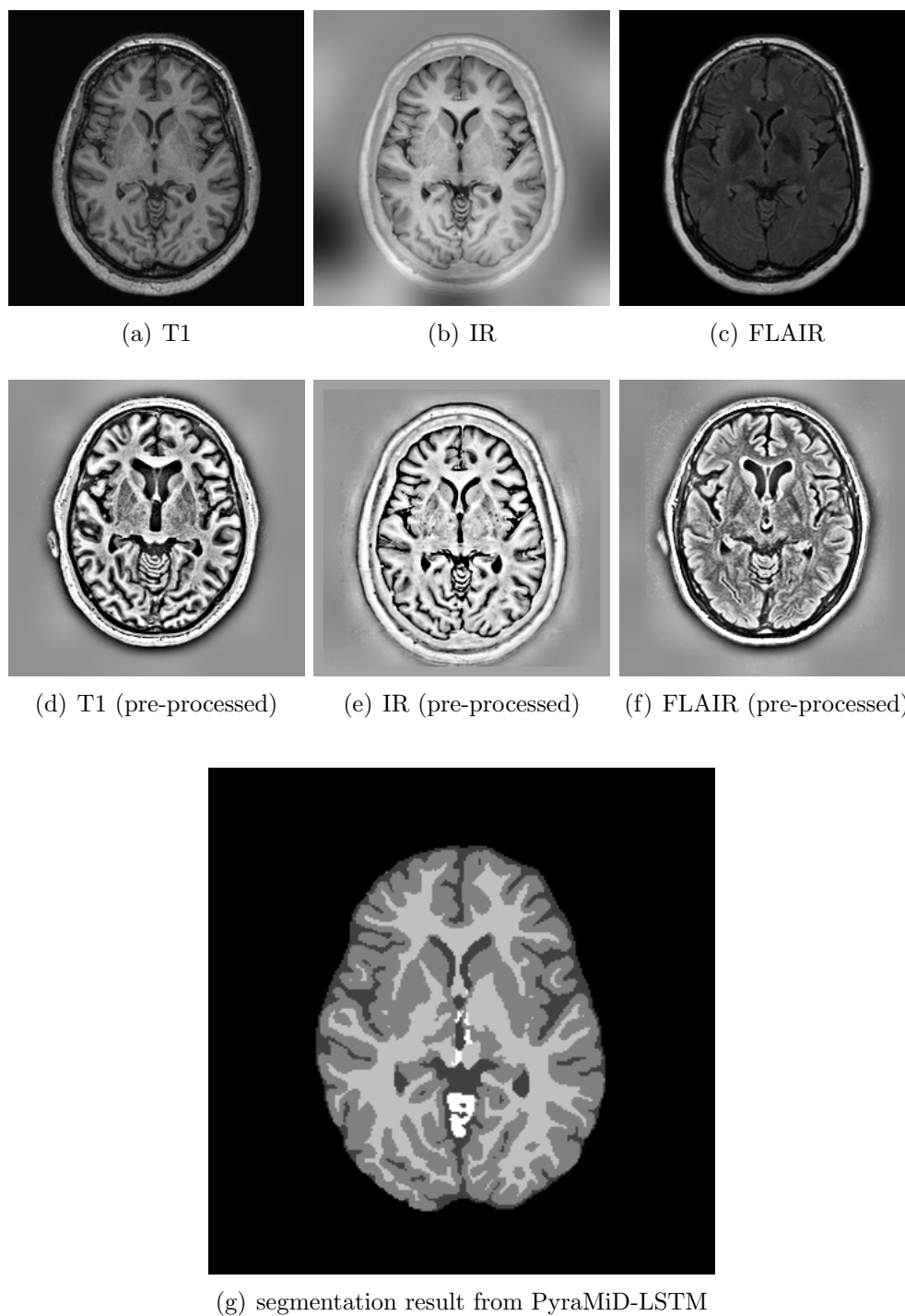


Figure 7.2: Slice 19 of the test image 1. (a)-(c) are examples of three scan methods used in the MR brain dataset, and (d)-(f) show the corresponding images after our pre-processing procedure (see pre-processing in Section 7.1.3). Input (b) is omitted due to strong artifacts in the data — the other datatypes are all used as input to the PyraMiD-LSTM. The segmentation result is shown in (g).

Chapter 8

Conclusion and Future Work

This thesis has presented an LSTM-based framework for solving the problem of image analysis, with a focus on image classification and segmentation. There exist a number of challenges regarding these tasks with a variety of changes in illumination, view point, scale, and so on. Furthermore, images taken from the web are very diverse in resolution, quality, and context with a lot of noise and clutter. Therefore, this thesis has designed a system that is capable of solving such challenges in an end-to-end manner based on the hypothesis; MD-LSTM leads to the comprehensive vision system using raw pixel values rather than selected features on complex real-world input data (see Section 1.3 for the details of the research hypothesis of this thesis).

The main contributions build upon ideas emerging from the field of CV and DL to provide a general approach for various types of images: texture images, natural scene images, and biomedical volumetric images. MD-LSTM has an elegant recursive way of taking each pixel's entire spatio-temporal context into account. The evaluations have shown that MD-LSTM provides promising results on those images, and that MD-LSTM outperforms the state-of-the-art approaches including deep CNNs, which supports the validity of the hypothesis of this study.

8.1 Concluding Remarks

This thesis has investigated the MD-LSTM model and its network architecture for various CV problems. The key contributions are: the architecture design of MD-LSTM networks for 2D and 3D images, the automatic image analysis framework for image classification and segmentation, and the advanced databases for texture segmentation and multi-class scene classification. In this section, the details of the key contributions and some remarks or limitations of each contribution are discussed.

Traditionally, *MD-LSTM* is connected with two self-hidden units along with the axes (x and y in 2D) that carry the neighboring context of each pixel. The main drawback of the traditional method is that it is hard to parallelize due to the dependencies between the pixels, especially when using higher dimensional data. Therefore, a new MD-LSTM architecture, *PyraMiD-LSTM*, is proposed, which solves the problem that MD-LSTM has. *PyraMiD-LSTM* changes the connection strategy that makes parallelization much easier. The implementation on GPU hardware is efficient and fast in dealing with large volumetric data.

With these two MD-LSTM models, various network design choices for image analysis are suggested. The major design choices are based on three main factors: (1) the dimension of input, (2) the depth of the network, and (3) the type of output.

First, regarding the dimension of input, this thesis mainly uses 2D and 3D data. For the efficiency issue, the traditional MD-LSTM is mainly applied for 2D data and *PyraMiD-LSTM* for 3D data. MD-LSTM has less computations but cannot parallelize. Instead, *PyraMiD-LSTM* can efficiently be parallelized and makes use of more effective computations for high-dimensional data.

The choice of depth of the network is based on the input and the task. A deeper network builds higher-level abstraction in each layer. However, it requires more parameters and is hard to train. Several experiments in this thesis show that a single-layer network with LSTM can be successfully applied to challenging texture and scene images. In addition, especially for high-resolution images and for the images including high intra-class variation, a stack of LSTM layers is combined with a fully connected layer to a deeper network. The fully connected layer between the LSTM layers controls the number of weight parameters and provides a feature mapping between the LSTM layers.

The type of output matters when designing the output layer. In this thesis, the main tasks consist of image classification (a single label per image) and image segmentation (a set of labels that is the same size as the number of input pixels). In general, the output from the last hidden layer (either from the LSTM or the fully connected layer) is the activation vector per pixel. For segmentation, this output is directly sent to the softmax layer to produce the class probabilities per pixel. However, for classification, the network needs to find the highest score of the class over all pixels that provide the final class label of an input image. Therefore, an additional layer, a collapse layer, combining all activations over all pixels, is added between the hidden layer and the output (softmax) layer. These architecture options are able to make LSTM possible to solve diverse CV problems. Furthermore, a number of network settings and generalization techniques for a MD-LSTM network are analyzed and evaluated. They include the input representation, the weight initialization, the peephole connection, the regularization techniques, the optimization techniques for the weight updates, and other hyper-parameter settings.

These network architectures mentioned above are evaluated on texture images, natural scene images, and biomedical volumetric images. A number of problems of image analysis are addressed by in this thesis: texture classification, scene understanding, texture segmentation, scene labeling, and 3D biomedical volumetric image segmentation.

The 2D-LSTM with the network for classification is evaluated on texture and scene images. With the texture data (*texture classification*), the network has been evaluated on five common benchmarks containing high intra-class variance (the variance within the class) and low inter-class variance (the variance between the classes). Although the classification on such data is very challenging, 2D-LSTM achieved the best performance on all datasets.

For *scene understanding*, each image contains multiple categories with a great amount of noise and unrelated contents. By performing two pruning rules on the output of the network, the system is able to produce the reliable multiple target output without any prior knowledge. This system is successfully applied to automatic web-image tagging. The images used in this experiment are collected from web-image search. The instance of each category is very diverse and images contain noise, clutter, and unrated contents. In some cases, the query word (keyword) does not match with the retrieved image—the web retrieval system is not accurate. Figure 8.1 shows some





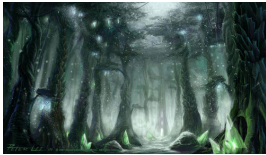
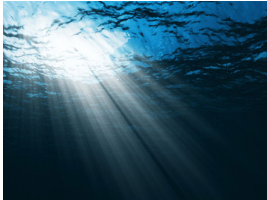
image	keyword tagging result	image	keyword tagging result
	flower sky, flower, candy		flower sky, flower, candy
	gravel building, stucco		forest flower, forest, ocean
	forest sky, forest, ocean		ocean ocean, sky

Figure 8.1: Examples of mislabeled web-images for tagging. Some examples show that the system can even improve the retrieval system.

examples of wrongly labeled images or images with ambiguous labels. The tagging results of the two images on the top with the keyword “flower” contain “candy”, which is a totally different object. However, it may be recognized by having a similar texture and color from the distance. The image with the keyword “gravel” (the left in the second row) has a large portion of a concrete rooftop. LSTM networks correctly classified the image as “building” and “stucco” which is more logical than the original keyword of the image “gravel”. The tagging results of the image with the keyword “forest” (the right in the second row) contain “ocean” which is not the correct classification. However, in some cases, the labels are not clearly visible in the image but are correct. For instance, in the left image in the last row, the image showing a forest under water is correctly tagged with the keyword “ocean” and “forest”. In the right one, “sky” is not visible, but the image was captured under the ocean with shafts of light (from the sky) cutting into the ocean. In this scenario, the logical answer is “ocean” with “sky”.

2D-LSTM with only a single hidden layer is investigated and evaluated for *texture segmentation*. Unlike other texture segmentation methods, LSTM precisely segments the related regions without any hand-crafted feature extractors. Here, the dataset is

challenging but the size of the images is pretty small; up to 100×100 .

To have more realistic scenarios, the LSTM is applied to a real-world scene images (*scene labeling*). Here, a deep network is employed to provide more robust classification accuracy with noisy natural images. However, there exists an issue on high-resolution scene images. The scene images contain large areas of background categories, such as sky and ground, but some foreground categories may appear among these backgrounds. In this case, the small regions of foreground tend to be forgotten and decayed by the large portion of background. Some examples of misclassification in such scenarios is shown in Figure 8.2. As can be seen, there are some mislabeled foreground categories which are labeled as a background.

Finally, a deep network with PyraMiD-LSTM is evaluated on volumetric data for *3D biomedical image segmentation*. The network with the GPU compatible implementation is successfully applied to two challenging benchmarks for biological volumetric image analysis and has outperformed other state-of-the-art methods.

To validate the framework on appropriate datasets, which is crucial for real-world application, a realistic dataset for each task is necessary. This thesis has introduced two challenging datasets for texture segmentation and scene understanding.

First, due to some limitations of current available texture segmentation datasets, an *Automated Blob-Mosaics Texture Dataset Generator* is proposed. It generates random 2D Gaussian blobs, where each blob is filled with random material textures with varying changes in illumination, pose, and scale. This dataset is challenging as it is hard to separate the related regions by human eyes.

The next dataset is generated for web-image analysis — *the web-scene image dataset*. The categories are decided based on some frequent and generic outdoor scenes (building, flower, forest, grass, snow, ocean, sand, sky, and gravel) and narrow in- and outdoor scenes (stucco, candy, and meat). The dataset consists of images collected from web-image search and offers different settings for training and testing. Images for training belong to a single category, so the network can be carefully trained with the individual categories. For testing, scene images are rich in context and highly variable in their variety of noise (watermarks, logos, or unrelated contents) and clutter. Furthermore, each image contains multiple categories that need to be classified.

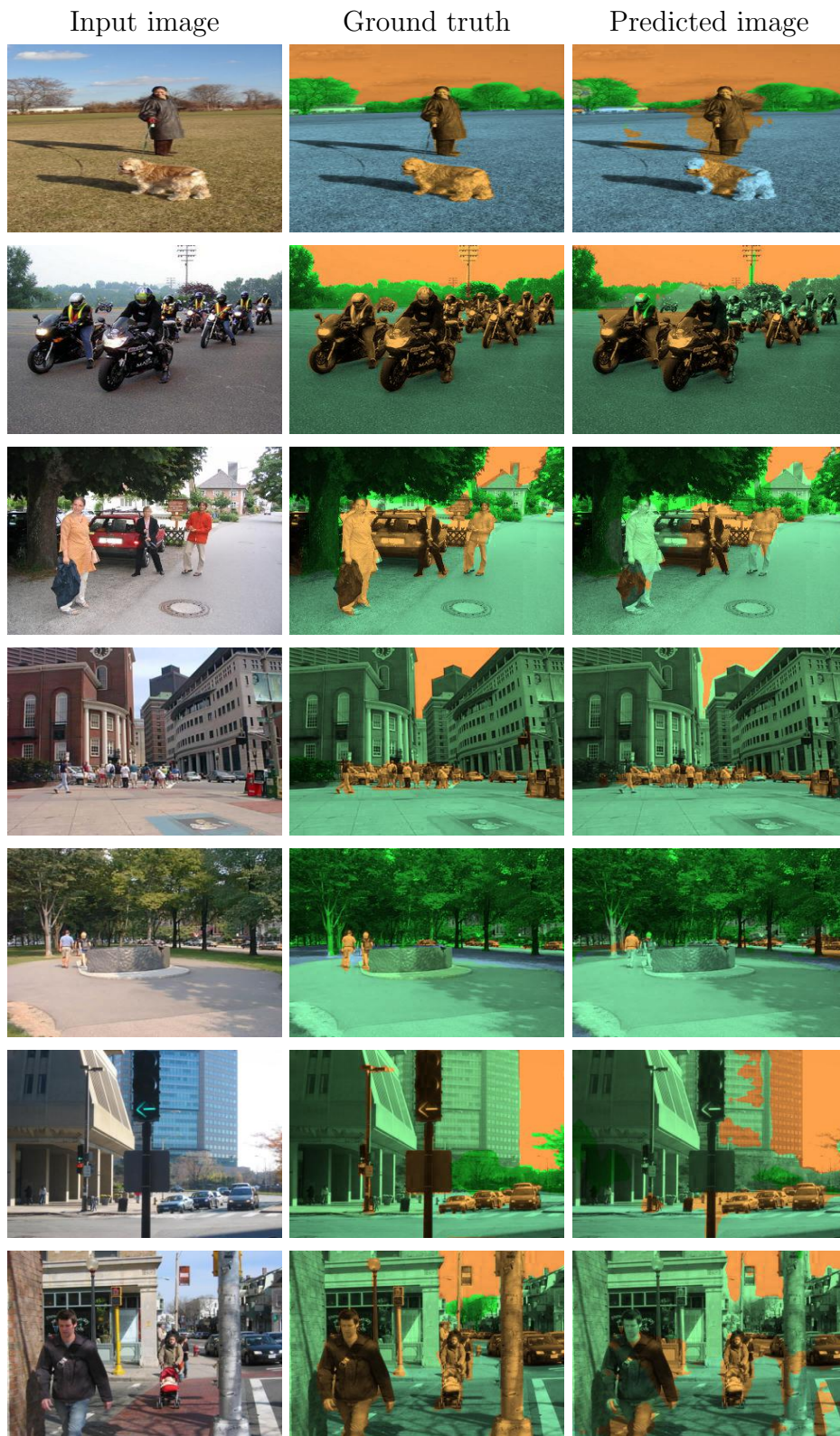


Figure 8.2: Examples of failing cases in scene labeling

This setting is challenging since multi-class image classification¹ is an open issue in CV communities.

8.2 Future Directions

Overall, the contributions in this thesis are major advancements in the direction of solving image analysis problems with LSTM without the need of any extra processing or manually designed steps. This section provides future directions and discusses some open issues in image analysis and the LSTM-based network. The aim of this thesis and the proposed directions is to improve the presented framework to achieve the ultimate goal of accurate fine-grained image analysis and human-like understanding of images by machines.

Reliable web-image tagging system: Due to time constraints, the size of the web-image dataset created here is not particularly large (12 categories, training: 350 attribute images per class, testing: 540 scene images). With learning-based algorithms, especially deep NNs, more training data is helpful to generalize the task and avoid overfitting. Furthermore, the number of categories is relatively small compared to other popular datasets of larger size, e.g., ImageNet (1000 categories, 1.2 million training images). We believe that the automatic web-image tagging system will be improved as image data grows.

Segmentation without target labels: Annotating images is one of the major issues in image segmentation. Note that the annotation is generally performed by humans, and manually performing the precise pixel-wise labeling needs a large effort. Therefore, generating image segmentation datasets with target labels is not easy. NN-based algorithms typically incorporate supervised learning. However, unsupervised learning² has recently become a popular direction in DL, e.g., Autoencoder. Another example, specifically designed for clustering, is Self Organizing Map (SOM), which is popularly applied for image segmentation. With this in mind, unsupervised learn-

¹An image is classified into more than two classes.

²Unsupervised learning is a type of ML algorithm which is trying to learn patterns from unlabeled data.

ing with a deep architecture can be a desirable direction for solving the annotation problem of image segmentation.

3D segmentation on videos: In Chapter 7, 3D volumetric segmentation is presented on biomedical images. As can be seen from the experiments, PyraMiD-LSTM or MD-LSTM provides an ability of taking the long-range spatio-temporal context of each pixel into account. Also, LSTM naturally brings the benefits of learning the context of multidimensional data. The assumption here is that PyraMiD-LSTM can be shifted across volumetric images or videos to segment them, since PyraMiD-LSTM was successfully applied to 3D volumetric data. PyraMiD-LSTM, thereby, can learn the dependencies in spatio-temporal domains, which is desirable for video analysis.

Bibliography

[A. 15] A. M. Mendrik, K. L. Vincken, H. J. Kuijf, G. J. Biessels, and M. A. Viergever (organizers). MRBrainS Challenge: Online Evaluation Framework for Brain Image Segmentation in 3T MRI Scans, <http://mrbrains13.isi.uu.nl>, 2015.

[ADBB04] Vincent Arvis, Christophe Debain, Michel Berducat, and Albert Benassi. Generalization of the cooccurrence matrix for colour images: application to colour texture classification. *Image Analysis and Stereology*, 23(1):63–72, 2004.

[AK11] Omar S Al-Kadi. Supervised texture segmentation: a comparative study. In *Applied Electrical Engineering and Computing Technologies (AEECT), 2011 IEEE Jordan Conference on*, pages 1–5. IEEE, 2011.

[AR02] Shivani Agarwal and Dan Roth. Learning a sparse representation for object detection. In Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen, editors, *Computer Vision ECCV 2002*, volume 2353 of *Lecture Notes in Computer Science*, pages 113–127. Springer Berlin Heidelberg, 2002.

[AV12] Susana Alvarez and Maria Vanrell. Texton theory revisited: A bag-of-words approach to combine textons. *Pattern Recognition*, 45(12):4312–4325, 2012.

[bar] NewbarkTex image test suite. https://www-lisic.univ-littoral.fr/~porebski/BarkTex_image_test_suite.html.

[BB14] Wonmin Byeon and Thomas M. Breuel. Supervised texture segmentation using 2d {LSTM} networks. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 4373–4377, Oct 2014.

- [BBRL15] Wonmin Byeon, Thomas M. Breuel, Federico Raue, and Marcus Liewicki. Scene labeling with {LSTM} recurrent neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, June 2015.
- [BCV13] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.
- [Bea78] P. R. Beaudet. Rotationally invariant image operators. In *International Conference on Pattern Recognition*, 1978.
- [Ben09] Yoshua Bengio. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127, January 2009.
- [BETG08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia.
- [BHSF11a] Francesco Bianconi, Richard Harvey, Paul Southam, and Antonio Fernández. Theoretical and experimental comparison of different approaches for color texture classification. *Journal of Electronic Imaging*, 20(4):043006–043006, 2011.
- [BHSF11b] Francesco Bianconi, Richard Harvey, Paul Southam, and Antonio Fernández. Theoretical and experimental comparison of different approaches for color texture classification. *Journal of Electronic Imaging*, 20(4):043006–043006–17, 2011.
- [BK04] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, Sept 2004.
- [BL07] Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards ai. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large-Scale Kernel Machines*. MIT Press, 2007.

- [BLB14] Wonmin Byeon, M. Liwicki, and T.M. Breuel. Texture classification using 2d {LSTM} networks. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 1144–1149, Aug 2014.
- [BLB15] Wonmin Byeon, Marcus Liwicki, and Thomas M. Breuel. Scene analysis by mid-level attribute learning using 2d {LSTM} networks and an application to web-image tagging. *Pattern Recognition Letters*, 63:23 – 29, 2015.
- [BMP02] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):509–522, Apr 2002.
- [BPA⁺09] Kolawole Oluwole Babalola, Brian Patenaude, Paul Aljabar, Julia Schnabel, David Kennedy, William Crum, Stephen Smith, Tim Cootes, Mark Jenkinson, and Daniel Rueckert. An evaluation of four automatic methods of segmenting the subcortical structures in the brain. *NeuroImage*, 47(4):1435 – 1447, 2009.
- [Bro66] P Brodatz. Textures: A photographic album for artists and designers dover publications. *New York, USA*, 1966.
- [BSF94] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.
- [BZM07a] A. Bosch, A. Zisserman, and X. Muoz. Image classification using random forests and ferns. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, Oct 2007.
- [BZM07b] Anna Bosch, Andrew Zisserman, and Xavier Muoz. Image classification using random forests and ferns. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.
- [Can86] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, Nov 1986.
- [CDF⁺04] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In

- Workshop on statistical learning in computer vision, ECCV*, volume 1, pages 1–2. Prague, 2004.
- [CG10] Michael Crosier and Lewis D Griffin. Using basic image features for texture classification. *International Journal of Computer Vision*, 88(3):447–460, 2010.
- [CGGS12] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2852–2860, 2012.
- [CMS12] D. C. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. Technical report, IDSIA, February 2012. arXiv:1202.2745v1 [cs.CV].
- [CPK⁺14] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *CoRR*, abs/1412.7062, 2014.
- [CSP⁺10] Albert Cardona, Stephan Saalfeld, Stephan Preibisch, Benjamin Schmid, Anchi Cheng, Jim Pulokas, Pavel Tomancak, and Volker Hartenstein. An integrated micro-and macroarchitectural analysis of the drosophila brain by computer-assisted serial section electron microscopy. *PLoS biology*, 8(10):e1000502, 2010.
- [CWV⁺14] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning. *CoRR*, abs/1410.0759, 2014.
- [DBLFF10] Jia Deng, AlexanderC. Berg, Kai Li, and Li Fei-Fei. What does classifying more than 10,000 image categories tell us? In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision ECCV 2010*, volume 6315 of *Lecture Notes in Computer Science*, pages 71–84. Springer Berlin Heidelberg, 2010.
- [Der87] Rachid Deriche. Using canny’s criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, 1(2):167–187, 1987.

- [DHS11] John Duchi, E Hazan, and Y Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning*, 12:2121–2159, 2011.
- [Dic45] Lee R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):pp. 297–302, 1945.
- [DS03] G. Dorko and C. Schmid. Selection of scale-invariant parts for object class recognition. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 634–639 vol.1, Oct 2003.
- [DT05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893 vol. 1, June 2005.
- [DW01] Alexandru Drimborean and Paul F Whelan. Experiments in colour texture analysis. *Pattern Recognition Letters*, 22(10):1161–1167, 2001.
- [DY14] Li Deng and Dong Yu. *Deep Learning: Methods and Applications*. NOW Publishers, 2014.
- [EBC⁺10] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, March 2010.
- [FCNL13] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1915–1929, Aug 2013.
- [Fle96] Arthur Flexer. Statistical evaluation of neural network experiments: Minimum requirements and current practice. *Cybernetics and Systems Research*, pages 1005–1008, 1996.
- [Fuk80] K. Fukushima. Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.
- [FZ07] Vittorio Ferrari and Andrew Zisserman. Learning visual attributes. In *Advances in Neural Information Processing Systems 20, Proceedings of*

the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007, 2007.

- [GBC] David Grangier, Lon Bottou, and Ronan Collobert. Deep convolutional networks for scene parsing.
- [GDDM14a] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587, June 2014.
- [GDDM14b] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587, June 2014.
- [Ger01] Felix Gers. Long short-term memory in recurrent neural networks, 2001.
- [GFK09a] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1–8, Sept 2009.
- [GFK09b] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1–8, Sept 2009.
- [GFS07a] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Multi-dimensional recurrent neural networks. In *Artificial Neural Networks–ICANN 2007*, pages 549–558. Springer, 2007.
- [GFS07b] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Multi-dimensional recurrent neural networks. In Joaquim Marques de S, LusA. Alexandre, Wodzisaw Duch, and Danilo Mandic, editors, *Artificial Neural Networks ICANN 2007*, volume 4668 of *Lecture Notes in Computer Science*, pages 549–558. Springer Berlin Heidelberg, 2007.
- [GLF⁺09] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for improved unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5), 2009.

- [Gra] Alex Graves. Rnnlib: A recurrent neural network library for sequence learning problems. <http://sourceforge.net/projects/rnnl/>.
- [Gra08] Alex Graves. *Supervised sequence labelling with recurrent neural networks*. PhD thesis, Technische Universitt Mnchen, 2008.
- [Gra12] Alex Graves. *Supervised sequence labelling with recurrent neural networks*, volume 385. Springer, 2012.
- [GRVG12] Juergen Gall, Nima Razavi, and Luc Van Gool. An introduction to random forests for multi-class object detection. In Frank Dellaert, Jan-Michael Frahm, Marc Pollefeys, Laura Leal-Taix, and Bodo Rosenhahn, editors, *Outdoor and Large-Scale Real-World Scene Analysis*, volume 7474 of *Lecture Notes in Computer Science*, pages 243–263. Springer Berlin Heidelberg, 2012.
- [GS08] Alex Graves and Juergen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 545–552, 2008.
- [GSC00] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000.
- [GSS02] F. A. Gers, N. Schraudolph, and J. Schmidhuber. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, 3:115–143, 2002.
- [Har79] Robert M Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, 1979.
- [HBFS01] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.
- [HKR93] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge. Comparing images using the hausdorff distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(9):850–863, Sep 1993.

- [HM08] Michal Haindl and Stanislav Mikes. Texture segmentation benchmark. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.
- [Hoc91] S. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991. Advisor: J. Schmidhuber.
- [HS88] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Citeseer, 1988.
- [HS97a] S Hochreiter and J Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, Nov 1997.
- [HS97b] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [HS06] Geoffrey Hinton and Ruslan Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.
- [HSK⁺12] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012. Technical Report arXiv:1207.0580.
- [HZCP04] Xuming He, R.S. Zemel, and M.A. Carreira-Perpindn. Multiscale conditional random fields for image labeling. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–695–II–702 Vol.2, June 2004.
- [JBR⁺10] V. Jain, B. Bollmann, M. Richardson, D.R. Berger, M.N. Helmstaedter, K.L. Briggman, W. Denk, J.B. Bowden, J.M. Mendenhall, W.C. Abraham, K.M. Harris, N. Kasthuri, K.J. Hayworth, R. Schalek, J.C. Tapia, J.W. Lichtman, and H.S. Seung. Boundary learning by optimization with topological constraints. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2488–2495, June 2010.
- [Jia13] Yangqing Jia. Caffe: An open source convolutional architecture for fast feature embedding. <http://caffe.berkeleyvision.org/>, 2013.

- [JXY13] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1):221–231, Jan 2013.
- [KDG15] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long short-term memory. *CoRR*, abs/1507.01526, 2015.
- [KEF⁺14] Taygun Kekeç, Rémi Emonet, Elisa Fromont, Alain Trémeau, Christian Wolf, and France Saint-Etienne. Contextually constrained deep networks for scene labeling. In *Proceedings of the British Machine Vision Conference, 2014*, 2014.
- [KK10] M.P. Kumar and D. Koller. Efficiently selecting regions for scene understanding. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3217–3224, June 2010.
- [KN12] L. Kratz and K. Nishino. Tracking pedestrians using local spatio-temporal motion patterns in extremely crowded scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(5):987–1002, May 2012.
- [KSH12a] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114, 2012.
- [KSH12b] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.
- [KSZ14] Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014.
- [KTH] KTH-TIPS and KTH-TIPS2 texture image database. <http://www.nada.kth.se/cvap/databases/kth-tips/download.html>.
- [KTS⁺14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural

- networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1725–1732, June 2014.
- [KWT88] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988.
- [LBBH98a] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.
- [LBBH98b] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [LBOM98] Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop. In G. Orr and Muller K., editors, *Neural Networks: Tricks of the trade*. Springer, 1998.
- [LCY13] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013.
- [LGTB97] S. Lawrence, C.L. Giles, Ah Chung Tsoi, and A.D. Back. Face recognition: a convolutional neural-network approach. *Neural Networks, IEEE Transactions on*, 8(1):98–113, Jan 1997.
- [Lin98] Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, 1998.
- [LJ08] D. Larlus and F. Jurie. Combining appearance models and markov random fields for category level object segmentation. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–7, June 2008.
- [LJST14] Ting Liu, Cory Jones, Mojtaba Seyedhosseini, and Tolga Tasdizen. A modular hierarchical approach to 3d electron microscopy image segmentation. *Journal of Neuroscience Methods*, 226(0):88 – 102, 2014.
- [LLS⁺15] Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, and Gang Hua. A convolutional neural network cascade for face detection. June 2015.

- [Low99] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157 vol.2, 1999.
- [Low04] DavidG. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.
- [LRSM13] Tian Lan, M. Raptis, L. Sigal, and G. Mori. From subcategories to visual composites: A multi-level framework for object detection. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 369–376, Dec 2013.
- [LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.
- [LSP06] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178, 2006.
- [LYT09] Ce Liu, J. Yuen, and A Torralba. Nonparametric scene parsing: Label transfer via dense scene alignment. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1972–1979, June 2009.
- [LYT11] Ce Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing via label transfer. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(12):2368–2382, Dec 2011.
- [LZCR15] Liang Lu, Xingxing Zhang, Kyunghyun Cho, and Steve Renals. A study of the recurrent neural network encoder-decoder for large vocabulary speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.
- [MCUP04] J Matas, O Chum, M Urban, and T Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761 – 767, 2004. British Machine Vision Computing 2002.

- [MLKS13] Roni Mittelman, Honglak Lee, Benjamin Kuipers, and Silvio Savarese. Weakly supervised learning of mid-level features with beta-bernoulli process restricted boltzmann machines. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 476–483. IEEE, 2013.
- [MM03] G. Mori and J. Malik. Recognizing objects in adversarial clutter: breaking a visual captcha. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I-134–I-141 vol.1, June 2003.
- [MOVY01] Bangalore S Manjunath, J-R Ohm, Vinod V Vasudevan, and Akio Yamada. Color and texture descriptors. *Circuits and Systems for Video Technology, IEEE Transactions on*, 11(6):703–715, 2001.
- [MS02] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen, editors, *Computer Vision ECCV 2002*, volume 2350 of *Lecture Notes in Computer Science*, pages 128–142. Springer Berlin Heidelberg, 2002.
- [MS04] Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.
- [MS05a] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630, Oct 2005.
- [MS05b] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630, 2005.
- [MVK⁺02] Christian Münzenmayer, Heiko Volk, Christian Küblbeck, Klaus Spinnler, and Thomas Wittenberg. Multispectral texture analysis using interplane sum-and difference-histograms. In *Pattern Recognition*, pages 42–49. Springer, 2002.
- [MZICS13] Shugao Ma, JianMing Zhang, N. Ikizler-Cinbis, and S. Sclaroff. Action recognition and localization by hierarchical space-time segments.

- In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2744–2751, Dec 2013.
- [NC13] C. Nieuwenhuis and D. Cremers. Spatially varying color distributions for interactive multilabel segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(5):1234–1247, May 2013.
- [NFF07] J.C. Niebles and Li Fei-Fei. A hierarchical model of shape and appearance for human action classification. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, June 2007.
- [OB14] Enrique G. Ortiz and Brian C. Becker. Face recognition for web-scale datasets. *Computer Vision and Image Understanding*, 118:153 – 170, 2014.
- [OFPA04] A. Opelt, M. Fussenegger, A. Pinz, and P. Auer. Weak hypotheses and boosting for generic object detection and recognition. In Toms Pajdla and Ji Matas, editors, *Computer Vision - ECCV 2004*, volume 3022 of *Lecture Notes in Computer Science*, pages 71–84. Springer Berlin Heidelberg, 2004.
- [out] OuTex image test suite. <http://www.outex.oulu.fi/index.php?page=classification>.
- [PAA⁺87] Stephen M. Pizer, E. Philip Amburn, John D. Austin, Robert Cromartie, Ari Geselowitz, Trey Greer, Bart Ter Haar Romeny, and John B. Zimmerman. Adaptive histogram equalization and its variations. *Comput. Vision Graph. Image Process.*, 39(3):355–368, September 1987.
- [Pal04] Christoph Palm. Color texture classification by integrative co-occurrence matrices. *Pattern Recognition*, 37(5):965–976, 2004.
- [PC14] Pedro Pinheiro and Ronan Collobert. Recurrent convolutional neural networks for scene labeling. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 82–90. JMLR Workshop and Conference Proceedings, 2014.

- [PD07] Florent Perronnin and Christopher Dance. Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.
- [PH12] G. Patterson and J. Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2751–2758, June 2012.
- [PL02] Christoph Palm and Thomas M Lehmann. Classification of color textures by gabor filtering. *Machine Graphics and Vision*, 11(2/3):195–220, 2002.
- [PP00] Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33, 2000.
- [PSCZ15] Tomas Pfister, Karen Simonyan, James Charles, and Andrew Zisserman. Deep convolutional neural networks for efficient pose estimation in gesture videos. In Daniel Cremers, Ian Reid, Hideo Saito, and Ming-Hsuan Yang, editors, *Computer Vision – ACCV 2014*, volume 9003 of *Lecture Notes in Computer Science*, pages 538–552. Springer International Publishing, 2015.
- [PVM07] Alice Porebski, Nicolas Vandenbroucke, and Ludovic Macaire. Iterative feature selection for color texture classification. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 3, pages III–509. IEEE, 2007.
- [PVM10] Alice Porebski, Nicolas Vandenbroucke, and Ludovic Macaire. Comparison of feature selection schemes for color texture classification. In *Image Processing Theory Tools and Applications (IPTA), 2010 2nd International Conference on*, pages 32–37. IEEE, 2010.
- [Ran71] William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

- [RST02] Remi Ronfard, Cordelia Schmid, and Bill Triggs. Learning to parse pictures of people. In Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen, editors, *Computer Vision ECCV 2002*, volume 2353 of *Lecture Notes in Computer Science*, pages 700–714. Springer Berlin Heidelberg, 2002.
- [RTK09] Chris Russell, Philip H. S. Torr, and Pushmeet Kohli. Associative hierarchical crfs for object class image segmentation. In *in Proc. ICCV*, 2009.
- [SBL15] Marijn F. Stollenga, Wonmin Byeon, Marcus Liwicki, and Juergen Schmidhuber. Parallel multi-dimensional {LSTM}, with application to fast biomedical volumetric image segmentation. In *Neural Information Processing Systems (NIPS), 2015 Conference on*, 2015.
- [Sch14] J. Schmidhuber. Deep learning in neural networks: An overview. Technical Report IDSIA-03-14 / arXiv:1404.7828v1 [cs.NE], The Swiss AI Lab IDSIA, 2014.
- [SCZ08] Florian Schroff, Antonio Criminisi, and Andrew Zisserman. Object class segmentation using random forests. In *BMVC*, pages 1–10, 2008.
- [Seg12] Segmentation of Neuronal Structures in EM Stacks Challenge. IEEE International Symposium on Biomedical Imaging (ISBI), <http://tinyurl.com/d2fgh7g>, 2012.
- [SEZ⁺13] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.
- [SHB⁺12] Richard Socher, Brody Huval, Bharath Bath, Christopher D Manning, and Andrew Y Ng. Convolutional-recursive deep learning for 3d object classification. In *Advances in Neural Information Processing Systems*, pages 665–673, 2012.
- [SL11] Pierre Sermanet and Yann LeCun. Traffic sign recognition with multi-scale convolutional networks. In *Proceedings of International Joint Conference on Neural Networks (IJCNN'11)*, pages 2809–2813, 2011.

- [Sla14] Thomas Grant Slatton. A comparison of dropout and weight decay for regularizing deep neural networks. In *An Undergraduate Honors College Thesis, University of Arkansas*, 2014.
- [SLJ⁺15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. June 2015.
- [SP97] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45:2673–2681, November 1997.
- [SP11] Jorge Sánchez and Florent Perronnin. High-dimensional signature compression for large-scale image classification. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1665–1672. IEEE, 2011.
- [SSB14] Hasim Sak, Andrew Senior, and Françoise Beaufays. Long Short-Term Memory recurrent neural network architectures for large scale acoustic modeling. In *Proc. Interspeech*, 2014.
- [SWRC06] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In Ale Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision ECCV 2006*, volume 3951 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin Heidelberg, 2006.
- [SyLNM11] Richard Socher, Cliff Chiung yu Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the International Conference on Machine Learning (ICML-11)*, 2011.
- [SZ14] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 568–576. Curran Associates, Inc., 2014.

- [TGJ⁺15] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. June 2015.
- [TH12] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4, 2012.
- [TL10] Joseph Tighe and Svetlana Lazebnik. Superparsing: Scalable nonparametric image parsing with superpixels. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision ECCV 2010*, volume 6315 of *Lecture Notes in Computer Science*, pages 352–365. Springer Berlin Heidelberg, 2010.
- [USC] The usc texture mosaic images. <http://sipi.usc.edu/database/database.php?volume=textures>.
- [VF10] Andrea Vedaldi and Brian Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the international conference on Multimedia*, pages 1469–1472. ACM, 2010.
- [visa] VisTexL image test suite. <http://www.outex.oulu.fi/index.php?page=contributed>.
- [visb] VisTexP texture image database. <http://vismod.media.mit.edu/vismod/imagery/VisionTexture/distribution.html>.
- [VKC⁺15] Francesco Visin, Kyle Kastner, Kyunghyun Cho, Matteo Matteucci, Aaron C. Courville, and Yoshua Bengio. Renet: A recurrent neural network based alternative to convolutional networks. *CoRR*, abs/1505.00393, 2015.
- [VTBE14] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *arXiv preprint arXiv:1411.4555*, 2014.
- [Wan12] Quan Wang. GMM-based hidden markov random field for color image and 3d volume segmentation. *CoRR*, abs/1212.4527, 2012.

- [WDB⁺08] Ofir Weber, Yohai S Devir, Alexander M Bronstein, Michael M Bronstein, and Ron Kimmel. Parallel algorithms for approximation of distance maps on parametric surfaces. *ACM Transactions on Graphics (TOG)*, 27(4):104, 2008.
- [WGS⁺15] Li Wang, Yaozong Gao, Feng Shi, Gang Li, John H. Gilmore, Weili Lin, and Dinggang Shen. Links: Learning-based multi-source integration framework for segmentation of infant brain images. *NeuroImage*, 108(0):160 – 172, 2015.
- [Wit84] Andrew P. Witkin. Scale-space filtering: A new approach to multi-scale description. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '84.*, volume 9, pages 150–153, Mar 1984.
- [WJWZ13] Shuo Wang, Jungseock Joo, Yizhou Wang, and Song-Chun Zhu. Weakly supervised learning for attribute localization in outdoor scenes. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3111–3118, June 2013.
- [WWZ13] Shuo Wang, Yizhou Wang, and Song-Chun Zhu. Hierarchical space tiling for scene modeling. In *Computer Vision–ACCV 2012*, pages 796–810. Springer, 2013.
- [WZZ⁺13] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L. Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In Sanjoy Dasgupta and David Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 1058–1066. JMLR Workshop and Conference Proceedings, May 2013.
- [XHE⁺10] Jianxiong Xiao, J. Hays, K.A Ehinger, A Oliva, and A Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3485–3492, June 2010.
- [ZF14] MatthewD. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision ECCV 2014*, volume 8689 of *Lecture Notes in Computer Science*, pages 818–833. Springer International Publishing, 2014.

- [ZJR⁺15] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. Conditional random fields as recurrent neural networks. *CoRR*, abs/1502.03240, 2015.
- [ZLZ13] Jun Zhang, Jimin Liang, and Heng Zhao. Local energy pattern for texture classification using self-adaptive quantization thresholds. 2013.
- [ZMLS07] Jianguo Zhang, Marcin Marszałek, Svetlana Lazebnik, and Cordelia Schmid. Local features and kernels for classification of texture and object categories: A comprehensive study. *International journal of computer vision*, 73(2):213–238, 2007.
- [ZZL12] Jun Zhang, Heng Zhao, and Jimin Liang. Continuous rotation invariant local descriptors for texon dictionary-based texture classification. *Computer Vision and Image Understanding*, 2012.

Curriculum Vitae

- Name: Wonmin Byeon
- Email: wonmin.byeon@gmail.com
- Education and Work Experience
 - Ph.D. in Computer Science
University of Kaiserslautern (TU KL), Germany, 2012 - 2016
Advisor: Prof. Thomas M. Breuel and Prof. Andreas Dengel
 - Visiting researcher
Dalle Molle Institute of Artificial Intelligence (IDSIA)
Swiss AI Lab, Artificial Intelligence team, Switzerland, 2015
Advisor: Prof. Juergen Schmidhuber
- Reserch Interest
 - Machine Learning
 - * Long Short-Term Memory Recurrent Neural Networks (LSTM)
 - * Convolutional Neural Networks (CNNs)
 - * Deep Learning
 - Computer Vision
 - * Image and video analysis
 - * Image/Video Classification and Segmentation