Cell Counting and Detection in Microscopy Images using Deep Neural Network

by

Yao Xue

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science
University of Alberta

Abstract

Microscopic image analysis is a broad term that covers the use of digital image processing techniques to process and analyze images obtained from a microscope. It is of significant interest to a number of diverse fields such as medicine, biological research, cancer research, drug testing, etc. A typical example is breast cancer, where the tumor proliferation speed (tumor growth) is an important biomarker indicative of the breast cancer patients condition. In practical scenarios, the most common method is to examine histological slides under a microscope based on pathologists' empirical assessments, which can be quite accurate in several cases, but generally is slow and prone to fatigue-induced errors. Among the above research fields, cell detection and cell counting are viewed as the central task or basis of microscopic image analysis. In this thesis, I address automatic cell detection and cell counting using computer vision and machine-learning methods.

Cell counting is to quantitate the population of specific cells from microscope images. The ability of accurate cell counting is important for precision diagnostics in laboratory medicine. I present a supervised learning framework with Convolutional Neural Network (CNN) and cast the cell counting task as an end-to-end regression framework, where the global cell count is taken as the annotation to supervise training, instead of following an object classification or detection framework. Compared to the idea of counting by detection, the regression framework evades the open and difficult problem of detection or segmentation of individual cells, and is more suitable for the counting task, whose end goal is to acquire the number of object instances. To further decrease the prediction error of counting, I fine-tune several cutting-edge CNN architectures (e.g., Deep Residual Network) into the regression model with Euclidean loss function rather than softmax loss function. As the final output, the proposed approach not only estimates the total number of certain cells in an image but also produces the spatial density prediction, which is able to describe the local cell density of an image sub-region. In many clinical imaging systems, researchers have confirmed that the topographic map that illustrates the cell density distribution is a valuable tool correlated with disease diagnose and treatment. The proposed method is evaluated with several state-of-the-art approaches on three cell image datasets and obtain superior performance.

As a related task to cell counting, cell detection focuses on localizing a certain type of cells or cellular subunits in microscopy images. Here, not only is the population of target cells of interest, but their locations in microscopy images are also valuable for subsequent biomedical research and clinical diagnosis. I propose a cell detection method based on Convolutional Neural Networks (CNNs) that uses encoding of the output pixel space. For the cell detection problem, the output space is the sparsely labeled pixel locations indicating cell centers. I employ random projections to encode the output space to a compressed vector of fixed dimensions. Then, CNN regresses this compressed vector from the input pixels. Furthermore, it is possible to stably recover sparse cell locations on the output pixel space from the predicted compressed vector using L_1-norm optimization. I conducted substantial experiments on several benchmark datasets, where the proposed CNN + CS framework (referred to as CNNCS) achieved the highestor at least top threeperformance in terms of F1-score, compared with other state-of-the-art methods.

On the basis of the proposed CNNCS model, I further develop an end-to-end trainable model, where the CNNCS model's two key components (a CNN-based regression model and CS-based sparse code predictor) are incorporated into a single network structure. Extensive experiments demonstrate the superior performance of the end-to-end trainable model on several challenging cell detection benchmark datasets.

Keywords: Cell Detection, Cell Counting, Convolutional Neural Network, Compressed Sensing, Microscopic Image Analysis.

Preface

Research for this thesis was conducted under the supervision of Dr. Nilanjan Ray at the University of Alberta. Portions of this thesis were published as:

- Chapter 3: Yao Xue, Nilanjan Ray, Judith Hugh, Gilbert Bigras. "Cell Counting by Regression Using Convolutional Neural Network." European Conference on Computer Vision (ECCV) workshop on Biomedical Imaging, pages: 274-290, 2016.
- Chapter 4: Yao Xue, Nilanjan Ray, Judith Hugh, Gilbert Bigras. "A Novel Framework to Integrate Convolutional Neural Network with Compressed Sensing for Cell Detection." IEEE International Conference on Image Processing (ICIP), pages: 2319-2323, 2017.
- Chapter 4: Yao Xue, Nilanjan Ray, Judith Hugh, Gilbert Bigras. "Output Encoding by Compressed Sensing for Cell Detection with Deep Convnet." Association for the Advancement of Artificial Intelligence (AAAI) workshop on Health Intelligence, 2018.
- Chapter 4: Yao Xue, Nilanjan Ray. "Cell Detection in Microscopy Images with Deep Convolutional Neural Network and Compressed Sensing." In arXiv preprint arXiv:1708.03307, 2017.

or are being under reviewed as:

- Chapter 4: Yao Xue, Judith Hugh, Gilbert Bigras, Nilanjan Ray. "Cell Detection in Microscopy Images with Deep Convolutional Neural Network and Compressed Sensing." IEEE Transactions on Medical Imaging, 2018.
- Chapter 5: Yao Xue, Judith Hugh, Gilbert Bigras, Nilanjan Ray. "End-to-End Learning of Output Encoding and Deep Convolutional Neural Network for Cell Detection." British Machine Vision Conference (BMVC), 2018.

Acknowledgements

My deep gratitude goes first to Professor Nilanjan Ray, my supervisor, for his constant encouragement and guidance. He has walked me through all the stages of writing this thesis and has given me generous support and helpful advice during these years of my PhD study. Without his consistent and illuminating instruction, this thesis could not have reached its present form. I believe that he has had a everlasting influence on my career.

Second, I would like to express my heartfelt gratitude to all my supervisor and committee members: Prof. Hong Zhang, Prof. Pierre Boulanger, Prof. Ron Kube, Prof. Faisal Z. Qureshi and Prof. Kumaradevan Punithakumar, who have taken their precious time off from their tight schedule, reading my thesis carefully and offering me constant encouragement, valuable suggestions and enlightening instructions, which contribute to the completion of my thesis.

I should also mention thanks to my past and present colleagues at the CIMS (Centre for IntelligentMining System) lab for providing me friendship, inspiration, advice and encouragement throughout my PhD. Furthermore, I extend my gratitude to Department of Computing Sciences for providing me with state of the art facilities and CSC (China Scholarship Council) for giving me the scholarship opportunity.

Last my thanks would go to my beloved family for their loving considerations and great confidence in me all through these years. I feel much grateful and heartily owe my achievement to my parents. I also owe my sincere gratitude to my friends and my fellow classmates who gave me their help and time in listening to me and helping me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Microscopy Image Analysis

Starting from the foundational theory of cells in the 19th century that found cells are the basic building blocks of life, biologists have strived to discover the hidden principles behind life. Significant progresses had been made across domains, but the demand to achieve more knowledge and explanations of cellular mechanisms has been increasing. Researchers want to manipulate cellular mechanisms to improve health, with more resources, effort, and modern equipment. As one of the most significant inventions, light microscopy [95] has promoted considerable research advances in cell biology ever since the 1670s when Antonie van Leeuwenhoek made the first attempt to apply microscopic imaging at the cellular level. In the present day, microscopic imaging techniques have enabled biologists to see phenomena much smaller than at the cellular level, for example, visualizing the transportation of chemicals or aggregation of proteins inside cells. Furthermore, microscopic imaging techniques have made it quite easy to obtain a huge amount of images within a few hours. The richness, dimensionality, and complexity of the microscopic images obtained by modern imaging equipment challenge manual image processing and analysis. As a result, automatic vision-based systems suitable for microscopic image processing and analysis have become indispensable and critical for proceeding advances in cell biology. Dating back to the mid-1950s, a system [89] was designed to automatically recognize smears of exfoliated cells, in order to perform identification of cervical cancer. This system was the first attempt to apply computers to the task of microscopic image processing and analysis. In the 1960s, [71] developed an automatic images processing scheme for counting the number of white blood cells in microscopic images. In their scheme, count estimation results are made by simple colorimetric and morphological measurements. During the mid-1970s, for the first time, a computer-assisted microscope [68] was designed to track and analyze morphological characteristics of neuronal cells. The arrival of confocal microscopes made 3D cell image analysis feasible in the 1980s. However up until the 1990s, computers had developed enough computation capacity to deal with 3D data and complex 2D data (e.g., histopathology [34]). At that time, the image processing and computer vision communities started to take over the task of microscopic image processing and analysis. After the year 2000, research papers on this topic have grown dramatically. Existing microscopic image analysis papers have become the foundation of cell-related research, including cell counting, cell detection, recognition of cell types, quantification of cell migration and interaction, cellular sociology or organization, and intracellular structures [75]. Among the aforementioned studies, cell detection and cell counting are often viewed as a central task or basis of microscopic image processing and analysis. The problems of cell detection and cell counting have received considerable attention in past years [84]. In this thesis, I have developed deep-learning-based cell detection and counting techniques that demonstrate excellent accuracies on benchmark datasets.

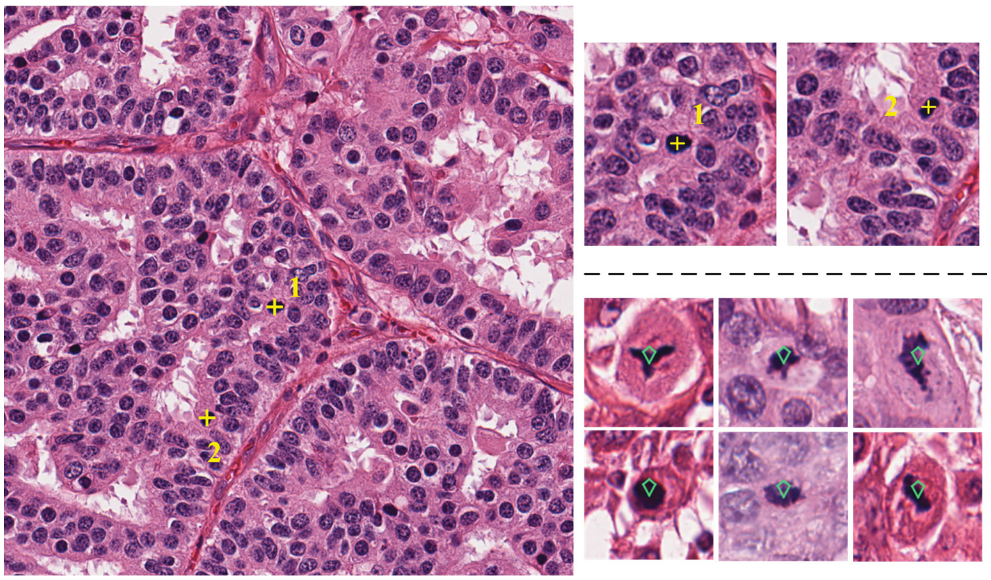


Figure 1.1: Left picture shows a microscopy image with two target cells annotated by yellow crosses at their centers. Right top pictures give details about the two target cells whose nuclei are in mitotic phase. Right bottom pictures provide more examples of mitotic figures.

1.2 Cell Detection

1.2.1 Significance and challenges

Automatic cell detection is used to find whether there are certain types of cells present in an input image (e.g., microscopy images) and to localize those in the image as shown in Figure 1.1. It is of significant interest to a wide range of medical imaging tasks and clinical applications. An example is breast cancer, where the tumor proliferation speed (tumor growth) is an important biomarker indicative of a breast cancer patients prognosis. In practical scenarios, the most common method is to examine histological slides under a microscope based on the pathologists' empirical assessments, which could be quite accurate in several cases, but is generally slow and prone to fatigue-induced errors. Manual analysis of microscopy images also suffers from inter-observer variabilities. In comparison, automatic and robust cell detection are highly desirable and serve as an essential prerequisite for a wide variety of subsequent tasks, such as cell segmentation, tracking, and recognition of cell types, etc. Numerous procedures in biology and medicine require cell counting and detection. For instance, a patients health can be inferred from the number of red blood cells and white blood cells. In clinical pathology, cell counts from images can be used for investigating hypotheses about developmental or pathological processes. And cell concentration is important in molecular biology, where it can be used to adjust the amount of chemicals to be applied in an experiment. In addition, the combination of cell detection and a successive stage of cell classification can provide clinically useful information about objects of interest, such as the presence (or quantity) of cancer cells in a microscopy image. The microscopy imaging market is significant and projected to exceed USD 5 billion in 2018[1]. Today every single microscope comes with a software suite that offers various types of automated image analysis, including detection of cells. However, these commercially available cell detection tools are not mature enough to serve the purpose of clinical diagnosis and applications. Recognizing the significance of cell detection and the inadequacy of available automated methods, medical imaging research communities have created challenging benchmark datasets and regularly organize challenges to keep track of the most recent progress in this area (see for example ICPR 2012 [77] and ICPR 2014 [2], MICCAI 2013 [94] and MICCAI 2016 [1] challenges).

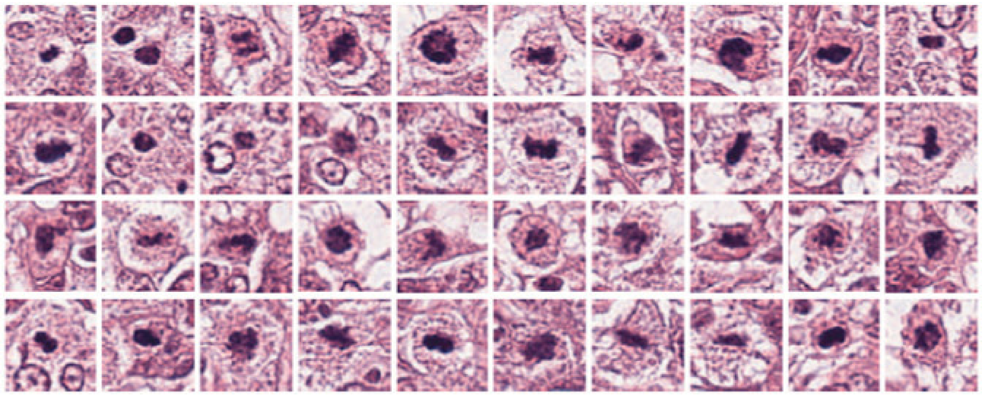


Figure 1.2: Example of breast cancer tumor cells.

Cell detection and localization constitute several challenges that deserve our attention. I enumerate some of these challenges below:

- Target cells are often surrounded by clutter represented by complex histological structures, such as capillaries, adipocytes, collagen, etc. It can be difficult to distinguish the target cells from the background clutter.

- There are also significant variations in appearance among the target cells themselves. Figure 1.2 visualizes a set of breast cancer tumor cells, which are all targets of interest

[1]`https://www.photonics.com/a55526/Trends_in_Microscopy_A_Big_Market_Focused_on`

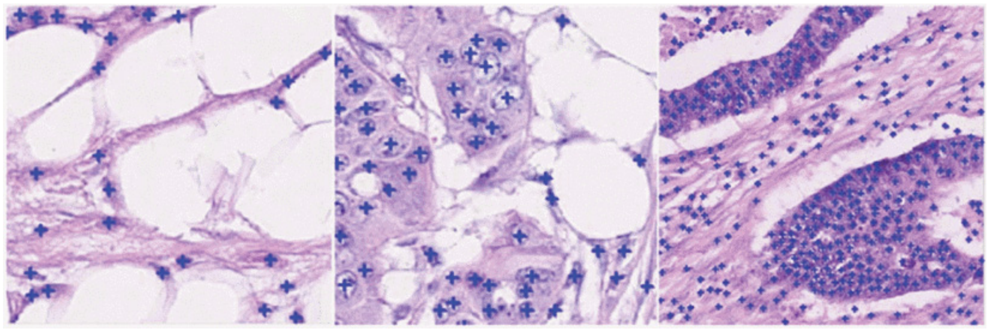


Figure 1.3: The density of target cells can vary a lot between images.

and present significant variations.

- Additionally, target cells can appear sparsely (only in tens), moderately densely (in tens of hundreds), or highly densely (in thousands) in a typical 2000 by 2000 pixel high resolution microscopy image, as shown in Figure 1.3.
- In many cases, the size of the target cell is small. It is unlike a general object detection problem where a computer system has to detect extended objects, e.g., human or vehicle detection.

These challenges render the cell detection and localization problems far from solved in spite of significant recent progress in computer vision research.

1.2.2 Limitations of existing automated cell detection methods

In the last few decades, different cell recognition methods have been proposed [27]. Traditional computer vision-based cell detection systems adopt a classical image processing pipeline, such as intensity thresholding, feature detection, morphological filtering, region accumulation, and deformable model fitting.

Classical machine-learning-based cell detection approaches offered superior accuracy over the aforementioned image processing pipeline. These methods follow a “hand-crafted feature representation” & “classifier” framework. First, the detection system extracts one or multiple features (e.g., HOG [17], LBP [70], LOG [51]) as the representation of input images. Then the method applies a classifier (such as Support Vector Machine) to the feature vector for cell detection. “Hand-crafted feature representation” & “classifier” approaches suffer from the following limitations:

1. It is a non-trivial and difficult task for humans to select suitable features. In many cases, it requires significant prior knowledge about the target cells and background.

2. Most hand-crafted features contain many parameters that are crucial for overall performance. Consequently, users need to perform a lot of trial-and-error experiments to tune these parameters.

3. Usually, one particular feature is not versatile enough. Features may often be tightly coupled with a particular type of target cell and may not work well when presented with a different type of target cell.

4. The performance of a hand-crafted feature-based classifier soon reaches an accuracy plateau, even when trained with plenty of training data.

Deep neural networks recently have been applied to a variety of computer vision problems, and have achieved better performance on several benchmark computer vision datasets [52], [32], [81]. The most compelling advantage of deep learning is that it has evolved from fixed-feature design strategies towards automated learning of problem-specific features directly from training data [54]. By providing massive amounts of training images and problem-specific labels, users do not have to go into the elaborate procedure of the extraction of features. Instead, Deep Neural Network (DNN) is optimized using a mini-batch gradient descent method over the training data, so that the DNN allows autonomic learning of implicit relationships within the data. For example, it has been demonstrated by several works [66] [106] that shallow layers of DNN focus on learning low-level features (such as edges, lines, dots), while deep layers of DNN form more abstract high-level semantic representation (such as probability maps, or object class category).

With the advent of deep learning in the computer vision community, it is no wonder that the state-of-the-art methods in cell detection are based on deep neural networks. The essential idea behind all of these methods is that detectors are trained as a classifier in the image pixel space, either as a pixel lableing [6] or as a form of region proposal network [13]. Thus, these methods predict the pixel coordinates of cells directly on a 2D image.

To explain the inadequacies of deep learning-based methods applied to cell detection, first note that cell detection from microscopy is a **point object** (such as cell centroid) detection problem. In comparison, object detection in computer vision refers to extended object detection with bounding boxes, such as detection of cars and humans among various other articulated objects that occupy a significant portion of the field of view. One could try to label cells with bounding boxes for training a deep learner; however, this labeling task may become nontrivial when a significant number of target cells are present with differing sizes. I argue that *general object detection methods with bounding boxes are not suitable for point object detection.* Thus, specially tailored methods need to be applied for cell detection.

State-of-the-art object detection categories fall into two groups: (a) methods dealing with a sliding window mechanism to cope with large pixel space, and/or, (b) discretizing the output space of bounding boxes and their locations. Region-based Convolutional Neural Network (R-CNN) [32] and its two variants Fast-RCNN [31] and Faster-RCNN [74] fall into the first category, while other significant methods, such as You Only Look Once (YOLO) [73] and Single Shot MultiBox Detector (SSD) [60] fall into the second category.

The RCNN family ([32], [35], [31], [74]) faces an overwhelming size of the output pixel space, because potentially every pixel is a target. The sliding window mechanism in these methods tends to produce false positives.

SSD [60] presented a method for detecting objects in images using a single deep neural network. SSD discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. Its assumption is that bounding boxes and locations are limited because of discretization. However, it then needs to regress on the entire pixel space for cell detection and is very likely to produce many false positives, like the RCNN family.

Similar in principle to the SSD method, YOLO [73] discretizes the pixel space into rather bigger patches and thus attempts to control the number of potential cell locations in the pixel space. However, larger discretization would be unsuitable for cell centroid detection because it would miss many cells.

Thus, I note that the source of the limitations of recent state-of-the-art general object detection applied to cell (centroid) detection is the combinatorial possibilities for cell locations on the entire pixel space. Therefore I argue that current general purpose object detectors are not designed to cope well with this enormous combinatorial output space.

The general purpose extended object detector would also run into another trouble, if cell detection is treated as a bounding box detection around small objects. For extended objects (such as humans, vehicles), localization is considered successful if a detection bounding box overlaps the actual bounding box by a threshold (e.g., 50%). For cell detection, the tolerance needs to be much tighter in order for the localization to be meaningful. Because in many cases, cells are quite small and also present huge variation in their density. Moreover, target cells are often only subtly different from other cells. Consequently, these methods tend to produce a significant amount of false positives.

There is yet another type of deep-learning-based methods (e.g, [6]) that explicitly try to tackle the combinatorial output space by converting the detection problem into a density prediction problem, where the modes of the density function serve as the point object (cell centroid) locations. Density prediction is possible by using fully convolutional architecture [81]; however, it has its own issues (e.g., nearby point objects can be merged to reduce the recall). On the other hand, if density function is not smooth enough, it may produce a significant number of false positives.

1.2.3 The proposed solution principle to cell centroid detection

I reiterate that cell detection from microscopy images is essentially a **point object detection problem.** An important consideration to solve point object detection is to overcome the curse of dimensionality, which I efficiently handle by applying random projections and regress on the reduced dimension. Reconstruction or restoration is possible from the reduced dimension because the final outputs (i.e., cell centroids) are quite sparse in the pixel space.

Additionally, our proposed method creates the opportunity for efficient ensemble averaging by introducing redundancies in a single neural network. The ensemble mechanism in our method enhances the accuracy of detections. Our method does not worry about merging nearby point objects, because regression is occurring in a transformed domain, not in the original pixel space. I also develop an end-to-end supervised deep learning framework to solve the cell centroid detection problem.

1.2.4 Outline of the proposed approach

In this section, I outline our proposed solution to cell detection in comparison with state-of-the-art cell detection methods. To explain the elements of novelty in our proposed cell detection solutions, I first present the workflows of two state-of-the-art methods [13] and [101] in the top two panels of Figure 1.4. It can be observed that the cell detectors of these state-of-the-art methods are trained in the image pixel space, either as a region proposal network [13] or as a spatial density predictor [101]. These methods strive to predict the pixel coordinates of cell centroids directly on a 2D image. Non-trivial post processing is inevitable to locate cell centroids.

The 3rd panel presents the proposed framework that integrates Convolutional Neural Network (CNN) with compressed sensing (CS) for cell detection and localization. Here, the CNN model predicts a fixed-length signal, which carries all the information about the true location of target cells. Then the CS-based sparse code predictor recovers sparse cell centroid locations. There are three principal reasons behind the conversion from detection in pixel space to regression in vector signal space. The first reason is technical. This conversion will let us turn the problem of detecting a variable number of targets into a fixed-length vector regression task, where one can apply state-of-the-art CNN architectures. Secondly, compared to pixel-level annotation of cells in image pixel space, a vector representation is more robust to inevitable system errors and can carry a set of redundant information, which provides us with the opportunity to do ensemble averaging to boost detection/localization performance. The third one is rather pragmatic; that one would strive to find an end-to-end training system that would make the training procedure simple and straightforward.

The training in the system (shown in the 3rd row) is occurring only within the CNN model. Thus, the sensing matrix in CS-based sparse code predictor is fixed, e.g., a random Gaussian matrix. This system is an intermediate framework before I migrate to an end-to-end framework, which is shown in the bottom panel of Figure 1.4. An end-to-end training system can back-propagate error signals from the very output end to all the previous modules in order to adjust the parameters of both the CNN model parameters and the sensing matrix in the CS module. It is in contrast with a conventional sequential pipeline, where each component is optimized independently and works one after another. The conventional sequential pipeline has several limitations. One of them is the process interdependence or error accumulation issue, which makes global optimization challenging. For example, a system consists of two modules, where the first module (e.g., a CNN) is trained with original

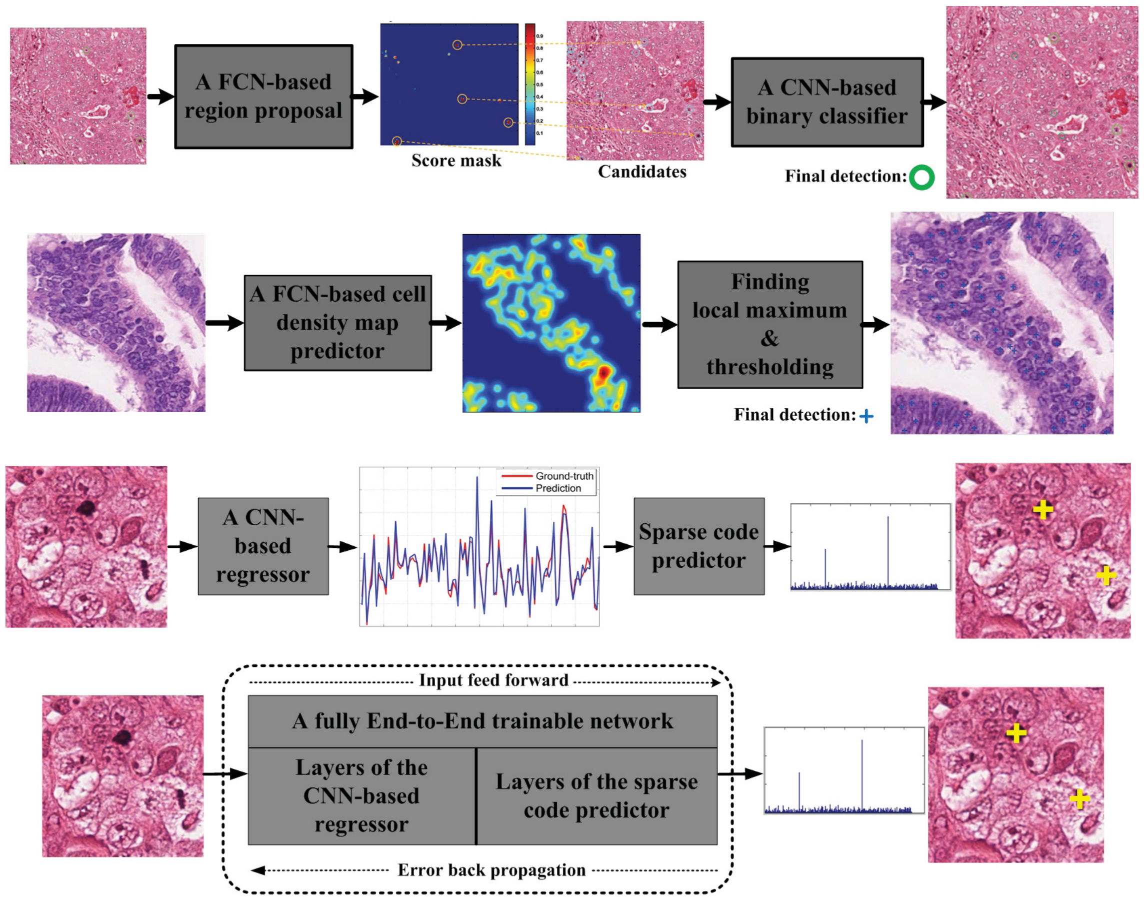

A FCN-based region proposal
Score mask
Candidates
A CNN-based binary classifier
Final detection:
A FCN-based cell density map predictor
Finding local maximum & thresholding
Final detection:
A CNN-based regressor
Ground-truth
Prediction
Sparse code predictor
Input feed forward
A fully End-to-End trainable network
Layers of the CNN-based regressor
Layers of the sparse code predictor
Error back propagation

Figure 1.4: The 1st and 2nd row show the diagram of two state-of-the-art methods [13] and [101], respectively, for cell detection. They represent the typical workflow of existing cell detection methods, which totally work in image pixel space and always strive to directly predict the x-y coordinate of target cells. The 3rd row gives the pipeline of the proposed CNNCS cell detection method [described in chapter-4]. The 4th row illustrates our End-to-End trainable version of the CNNCS method [described in chapter-5].

input data, and the second module depends on the output of the first module. As I know, the output of the first module is only an estimated result under certain loss functions or optimization policies. Thus the second module cannot give optimal results due to error accumulation, even though the second one is also trained to converge.

1.3 Cell Counting

1.3.1 Problem definition and significance

Automatic cell counting is to estimate the number of certain types of cells in microscopic images. It is of great interest to a wide range of medical scenarios [16] [79] [76]. An example is the diagnosis and treatment of breast cancer, which is one of the most common diseases leading to death among the female population worldwide. The number of proliferating (e.g., Ki67 positive) tumor cells is an important index associated with the severity of disease clinically. One available method of quantization involves counting the nuclei of proliferating cells using traditional image analysis techniques on a microscopic image. However, it has been proven to be challenging because of the inability to distinguish tumor cells from surrounding normal tissue like vessels, fat, and fibrous tissue [28], especially because in reality the resolution of the input medical image could be very high, at the same time that the target cells could easily be extremely dense. Consequently, it is quite difficult to manually count target cells one by one. This is the principal motivation for automatic cell counting.

1.3.2 Existing approaches and their limitations

Many traditional methods address counting tasks in an unsupervised manner, conducting grouping according to self-resemblances [61] or motion resemblances [72]. The counting precision of these fully unsupervised approaches is finite, and consequently alternative methods are proposed from the perspective of supervised learning. Following the success of deep learning applied in computer vision like detection, segmentation, and localization [52] [38] [36], the most recent cell counting-by-detection works choose a learning-based approach, where each training object has annotation information, which is sometimes dot annotation indicating the centroid of the object [15] [59] [101], and sometimes bounding-box annotation around the object [97]. However, it is well known that the problem of detecting and localizing individual object instances is far from being solved, especially in the real-world application of cell counting where cell density can be extremely high [53] [14] [91]. For example, the number of cells can easily reach or exceed thousands per image, and the cells also show huge variations in terms of type, size, shape, and appearance, etc.

Another related work is Fully Convolutional Neural Network (FCNN) [81], which has produced remarkable results in semantic segmentation and spatial density prediction. (In some ways, object counting can be seen as an integration over spatial density prediction.) To build the end-to-end and pixel-to-pixel FCNN, its training phase requires pixel-wise annotation, which is strongly supervised information and gives much benefit to FCNN. Consequently, work like [18] has been proposed.

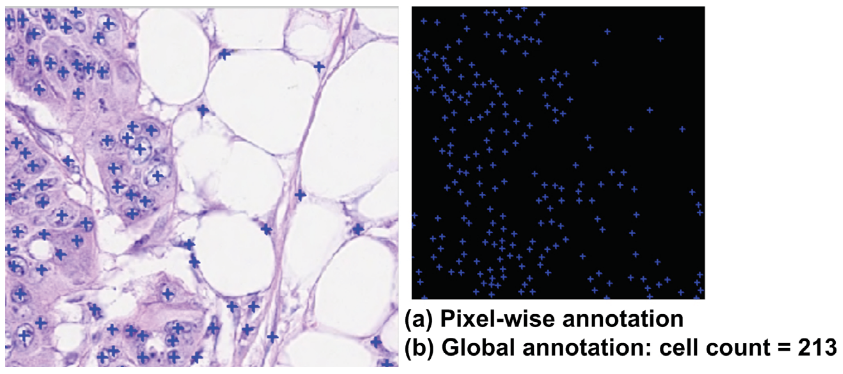

(a) Pixel-wise annotation
(b) Global annotation: cell count = 213

Figure 1.5: Counting-by-detection framework, most of which requires the pixel-wise annotation, is not a good choice in cell counting task, where cells could be extremely dense.

Figure 1.5 shows a cell image example with its two levels of annotations: (a) pixel-wise annotation, where the nuclei of each cell is dot-annotate, and (b) global count, where the total number of cell in the image is provided. Most counting-by-detection frameworks take the strongly supervised pixel-wise annotation as input during training, and then generate a global count (i.e., the total number of target cells) for the test image. However, the automatic cell counting task is to predict a global count only, thus it may be unnecessary to design an object-counting framework relying on the more expensive annotation data.

The problem of object counting from an image can be addressed from two distinct directions: 1) developing an object detector, and 2) developing an object counter. These two directions can be summarized as cell counting by detection and cell counting by regression.

Counting by detection

Counting by detection methods depend on vision-based object detectors, which are designed to look for every target object instance within an image. After the detection of target instances, object counting turns into a simple task. Many methods have chosen to fulfill object counting task following the detection pipeline [99] [102] [98] [6] [91] [58]. In this case, an object detection framework is designed to localize each object (e.g., cell, head, or vehicle) one by one; after that, a counter naturally takes all the detected objects and produces the final count. Following the success of deep learning applied in computer vision like detection, segmentation, and localization [52] [38] [36], most recent counting-by-detection works choose a learning-based approach, where each training object has annotation information, which is sometimes dot annotation indicating the centroid of the object [15] [59] [101], and sometimes bounding-box annotation around the object [97]. However, it is well known that the problem of detecting and localizing individual object instance is far from being solved, especially in real world application of cell counting, where cells are quite densely distributed in microscopic images [53] [14] [91]. For example, the number of cells can easily reach or exceed thousands per image, and the cells also show huge variations in terms of type, size, shape, and appearance, etc. Approaches within this direction are able to provide precise count estimation, if their fundamental suppositions are satisfied. However in general, these counting schemes do not become reliable enough when confronted with more challenging circumstances.

Counting by regression

Counting by regression approaches evade the open and difficult problem of detection or segmentation of individual cells. Efforts are made to learn the relationship from image representations (mainly various image global features) to the number of target instances. And a wide range of machine learning models (e.g., neural networks [50] [50] [64]) are good choices for solving such a standard regression problem. These methods abandon the known information regarding the exact position of targets, only relying on the target count for training.

Counting by density prediction [101] [57] can be viewed as a variant of counting by regression, since object counting can be fulfilled by an integration over the spatial density prediction. Counting by density prediction [101] formulated the task of cell counting as a supervised learning process, which strives to learn the relationship between an input image and its corresponding cell density map. The mechanism behind [101] is that cell density prediction evades the difficult problem of detection or segmentation of individual cells, and it is a good option for the cell counting task where only the quantity of cells is the demand. Another relevant work is [57], which designed a novel supervised learning scheme for a vision-based counting object, for example, predicting the number of cells. Similar to [101], the problem of cell counting is cast as an approximation of a cell density map, over which an integral can provide the number of target cells of the corresponding image so that it can also avoid the difficult mission of learning to detect or localize individual cells. It has been confirmed by several recent works [57] [101] that counting by regression is able to provide more efficient and precise cell counting results than counting by detection methods.

1.3.3 Proposed solution to cell counting

It is necessary to mention that several counting by regression methods apply deep-neural-network-based approaches, but their deep neural networks are usually trained and used under multi-class classification structures. Thus, the counting problem is cast as a classification problem, where the cell count is treated as class ID, and images with the same number of objects are seen as belonging to the same class. During its test phase, each test image is predicted with an integer class label, which indicates the number of objects in the image. However as I know from the object classification task, training images of different categories (for example, cat, bicycle, and airplane) are independent of each other, and softmax loss function is used in classification CNN architectures [52] [44] [36]. A classification-orientated CNN model treats cell counts 25 and 53 as far apart as counts 25 and 26. However from the nature of cell counting, the distance between different cell counts is important. It is beneficial to treat the object counting task as a regression task [91] where counts 25 and 26 should be closer, and that could be correctly reflected in the regression setting.

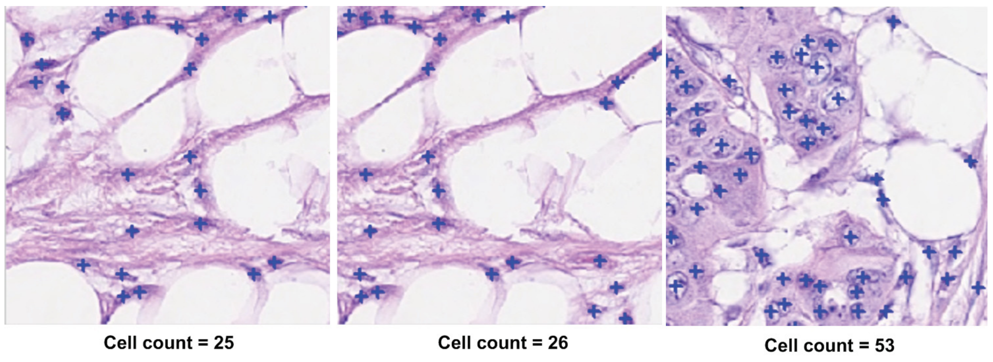


Figure 1.6: Cell counting is better modeled as a regression problem than a classification problem.

In our proposed solution to cell detection, I combine CNN-based regression with density estimation for cell counting. The schematic of our method is shown in Chapter 3, Figure 3.1. Thus, it is a combination of counting by detection and counting by regression methods. The advantages of proposed cell counting can be summarized as follows:

1. To capture the relationship between RGB cell image and its overall cell count, I cast the cell counting task as developing an end-to-end regression framework, which is more suitable for a counting task compared to counting by detection. Additionally, instead of being applied for classification purposes, a convolutional neural network architecture with Euclidean loss function is used for regression.

2. As the final output, the proposed approach not only estimates the global number of certain cells in an image but also produces the spatial density prediction, which is able to describe the local cell density of an image sub-region. In many clinical imaging systems [88] [24] [104], researchers have confirmed that the topographic map that illustrates cell density distribution is a valuable tool correlated with disease diagnose and treatment.

3. I utilize several popular CNN architectures (including the Deep Residual Network [36], AlexNet [52]) into our regression model. To the best of our knowledge, this is the first piece of work to expand the deep residual network from classification, detection, and segmentation to microscopy cell counting.

1.4 Organization and Contribution of the Thesis

Considering the nature of cell detection and localization tasks (for example, point objects, variable number of target cells, small size, sparse/high density, etc.), it is challenging to adapt classic object detection algorithms to cell detection. In this thesis, I integrate Compressed Sensing (CS) into a Convolutional Neural Network (CNN)-based detection framework to overcome these challenges. Chapter 4 describes the pipeline of the proposed CNNCS cell detection method. Chapter 5 describes the end-to-end trainable version of the CNNCS method.

Furthermore, a framework is also developed for counting target cells present in a microscopy image. Our cell counting framework is described in chapter 3. While in chapter 2, all the background and related work on general object detection, cell detection, cell counting, and compressed sensing theory is discussed.

The contribution of the proposed research can be summarized as follows.

- I cast the cell counting task as developing an end-to-end regression model, instead of following the object classification or detection framework. The regression framework evades the open and difficult problem of detection or segmentation of individual cells, and is more suitable for the counting task. To further decrease the prediction error of counting, I incorporate several cutting-edge CNN architectures (e.g., Deep Residual Network) into the regression model with Euclidean loss function rather than softmax loss function. The proposed method is evaluated with several state-of-the-art approaches on three cell image datasets and obtains superior performance.

- It is the first attempt demonstrating that deep convolutional neural network can work in conjunction with compressed sensing-based output encoding schemes toward solving a significant medical image processing task: cell detection and localization from microscopy images.

- I introduce redundancies in the CS-based output encoding that are exploited by CNN to boost generalization accuracy in cell detection and localization. These redundancies also help to reduce false detections.

- I demonstrate that the proposed CNNCS framework achieves competitive results compared to the state-of-the-art methods on several benchmark datasets and challenging cell detection contests.

- On the basis of the proposed CNNCS model, I further develop an end-to-end trainable model, which enables parameters of CNNCS's two key components being optimized jointly.

- Furthermore, it is the first work that derives a back propagation rule for a sparse coding (i.e., compressed sensing) algorithm. Our back propagation rule is independent of a sparse coding/compressed sensing algorithm. Also, our experiments on benchmark datasets show that our back propagation method increases accuracy by virtue of end-to-end training.

Chapter 2

Literature Survey

2.1 Cell Counting

Automated counting of cells from microscopy images or videos is a highly laborious and time-consuming task experienced by many real-world applications. A great number of processes in medicine and biology need cell counting; for example, a patient's physical condition can be deduced based on their quantity of red blood cells and white blood cells. In clinical pathology, cell counts in microscopy images are quite helpful to infer the pathological conditions or procedures of a patient, while in molecular biology, cell density is a key reference for regulating the concentration of chemicals used in an experiment.

Some traditional methods address counting tasks in an unsupervised manner, conducting grouping according to self-resemblances [61] or motion resemblances [72]. The counting precision of these fully unsupervised approaches is rather limited, and consequently alternative methods are proposed from the perspective of supervised learning. Following the success of deep learning applied in computer vision tasks, most recent methods (e.g., [57] [101] [59], etc.) choose deep-neural-network-based approaches to count cells present within an image.

Flaccavento et al. [30] have developed a learning algorithm that counts the number of cells in an image with a large field of view automatically, and can be used to investigate colony growth in time lapse sequences. The images are acquired using a novel, small, and cost-effective diffraction device that can be placed in an incubator during acquisition. This device, termed a CyMap, contains a resonant cavity LED and CMOS camera with no additional optical components or lenses. The counting method is based on structured output learning, and involves segmentation and computation using a random forest. They show that the algorithm can accurately count thousands of cells in a time suitable for immediate analysis of time lapse sequences.

More recent cell counting methods can be summarized as one of two directions. The first direction is cell counting by detection (e.g., [6]), which strives to segment or localize cells as the first step of their approach. The second direction is cell counting by regression, where there is no need for conducting prior cell detection or segmentation. For example, density prediction methods [7] [29] [57] are representative research within this direction.

Counting by detection

This direction relies on a vision-based object detector, which is designed to search for every target example within an image. After the detection of all target examples, object counting turns into a simple task. Nevertheless, it is well known that object detection is still far from being solved, particularly when cells are quite densely distributed in microscopic images. For example, most present object detectors work by two phases: first generating real-valued probability maps; then over each probability map, thresholding and non maximum suppression operation is required in order to localize high confidence points or regions corresponding to dense target examples [19] [69]. Other methods try to evade non-maximum suppression by inferring the mapping relationship between parts and examples [8] [22] [56] [100] [107], but they are only reliable for circumstances where there are a small number of objects in images, and they still need time-consuming reasoning processes. On the other hand, some approaches believe that it is very likely that objects are separable according to their different background appearances. As a result, the MonteCarlo process [20] and morphological analysis [5] are feasible here for detection of every single target example. Counting by detection-based methods is able to estimate object count accurately, when their basic hypothesis is satisfied. But generally speaking, counting by detection becomes not reliable enough once meeting with more challenging situations.

The method in [6] presented a machine-learning-based solution to the cell counting problem. The approach contains three major steps. First, it utilizes a fast Maximally Stable Extremal Regions (MSER) detector [65] to look for a large set of cell-like candidate regions. Second, a structured SVM-based model is trained for evaluating every candidate region by assigning a score in terms of its cell appearance. Finally, the non-overlapping regions with high confidence scores are chosen by dynamic programming.

Counting by regression

Counting by regression-based methods directly predicts the number of target instances from image representations (mainly various image global features), instead of paying attention to the open problem of object detection. A variety of machine-learning approaches (e.g., neural networks [50] [50] [64]) are available tools for the standard regression problem. These approaches rely on the target count for training instead of the pixel location coordinates of targets. Counting by segmentation approaches [12] [78] can be viewed as mixed breeds of counting-by-detection and counting-by-regression methods since objects are first segmented. Based on the global segmentation characteristics, the overall object counts are regressed.

A fully convolutional neural network (FCN) [81] was proposed for the image segmentation problem and had shown remarkable performance. Soon after the FCN was proposed, [101] developed a FCN-based framework (as shown in Figure 2.1) for cell counting, where their FCN is responsible for predicting a spatial density map of target cells, and then the number of cells can be estimated by an integration over the learned density map. [101]

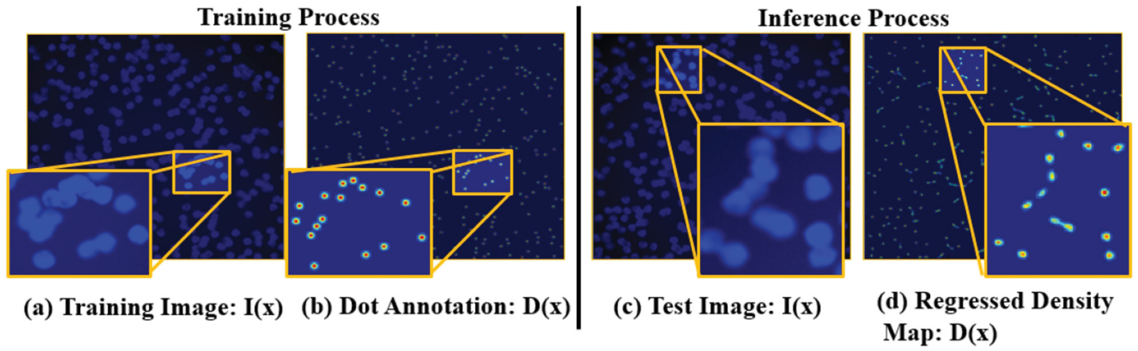


Figure 2.1: An overview of the FCN based framework for cell counting proposed in [101]. (a): Training image. (b): Dot annotations that create a Gaussian at the center of each cell. (c): Image from the test set. (d): Estimated Density Map, the number of cells in a specific region is calculated by integrating the density map over that region.

formulated the task of cell counting as a supervised learning process, which strives to learn the relationship between an input image and its corresponding cell density map. The mechanisms behind [101] is that cell density prediction evades the difficult problem of detection or segmentation of individual cells, and is a good option for cell counting tasks where only the quantity of cells is required.

The method in [57] designed a novel supervised learning scheme for vision-based counting object, for example, predicting the number of cells. Similar to [101], the problem of cell counting is cast as an approximation of a cell density map, over which the integral can provide the number of target cells of the corresponding image. By doing so they can also avoid the difficult mission of learning to detect or localize individual cells. Given an image I, the central idea of [57] is to learn a cell density function F of cell counts/square area in the image. The method assumes that every pixel in an image is described by a feature vector and formulates the density function as a linear transformation of the feature vector. Given a set of training images, the density function parameter is optimized, so that the density predicted for the training images resembles the ground truth density generated from the user annotations. It has been confirmed by several recent works [57] [101] that counting by regression is able to provide more efficient and precise cell counting results than those by detection methods.

2.2 Cell Detection and Localization

Automatic cell detection is of interest to a wide collection of cell-related research, since it is the foundation of many automatic studies ranging from cell type identification, quantification of cell migration, to intracellular structures. The broad variation of cells and microscopy imaging mechanisms ask for wide feasibility and generalization of cell detection methods to a variety of scenarios. The task of cell detection also becomes more difficult when the target cells becomes more dense, since cell clumping becomes quite common and cell sizes can change significantly. Furthermore, in many cases, both target cells and similar structures are simultaneously present in a microscopic image. It becomes harder for classical image processing techniques to still recognize the cells of interest.

Prior to the introduction of deep learning methods, cell detection and localization depended on segmentation of cells. An effective summary can be found in [27]. Traditional computer vision-based cell detection systems adopt classical image processing techniques, such as intensity thresholding, feature detection, morphological filtering, region accumulation, and deformable model fitting. For example, Laplacian-of-Gaussian (LoG) [51] operator was a popular choice for blob detection; Gabor filter or LBP feature [70] offers many interesting texture properties and had been attempted for a cell detection task [4]. Conventional cell detection approaches follow the "hand-crafted feature representation"+"classifier" framework.

Early research on automatic cell detection utilized domain-specific handcrafted features to build representations of statistical, morphological, or textural characteristics of mitosis [83] [49] [41] [93] [62] [96] [87]. Some of these works integrate two or more such features so as to achieve high detection accuracy. For instance, [83] developed a pixel-wise classifier using shape and texture features to localize target cells within a histology image. [41] designed a system that combined statistics and morphological features, and then performed joint analysis of the features using a decision tree classifier for mitosis detection. [87] distinguished mitotic and non-mitotic areas according to general features and a combination of cascaded adaboosts; it requires significant skills and prior knowledge to develop or choose appropriate handcrafted features. Furthermore, in many cases, such features are not effective enough to work as the representations of the properties of target cells, considering huge variations in shapes and textures, consequently leading to a low detection accuracy.

Today, the state-of-the-art methods in detection and localization include deep learning techniques for cell detection and localization. Recently, [101] presented a FCN-based framework for cell detection, where their FCN is responsible for predicting a spatial density map of target cells, and the local maximum point on the learned density map is considered as target cells. In another application of deep learning, a cascaded network [13] has been proposed for cell detection. [13] uses a FCN for candidate region selection, and then a CNN for further discrimination between target cells and background. In contrast to approaches using hand-crafted features, deep CNNs are able to learn hierarchical feature representations, and have achieved significant accuracy in object recognition tasks [52] [85] [82].

In cell detection, [15] employed a deep CNN as a pixel-wise classifier followed by ad-hoc post-processing to detect mitosis, and obtained the best performance in the 2012 ICPR MITOSIS challenge with an F1 score of 78.2%, and 61.1% in the 2013 MICCAI challenge. In [15], deep max-pooling convolutional neural networks were used to detect mitosis in breast histology images. The networks were trained to classify each pixel in the images, using a patch centered on the pixel as context. Afterward, post-processing was applied to the network output. But the deep CNN-based pixel-wise classifier is quite computationally-intensive. For a single microscopic image containing hundreds to thousands of high-power fields (HPFs), it is extremely time-consuming for the pixel-wise classifier to go through all sub-windows for detection. This greatly limits its application in clinical practice. In [80], expectation maximization has been utilized within a deep learning framework in an end-to-end fashion for mitosis detection. This work presents a new concept for learning from crowds that handles data aggregation directly as part of the learning process of the convolutional neural network (CNN) via an additional crowd-sourcing layer. It is the first work where deep learning has been applied to generate a ground-truth labeling from non-expert crowd annotation in a biomedical context.

As a very practical sub-problem of cell detection, mitoses detection in breast histology images has become a hot topic in recent years, and several automatic approaches have been proposed. Initial works utilized hand-crafted features that are dedicated to certain properties of mitosis for automatic detection [83] [49] [41] [93] [62] [96] [87]. But in many cases, these approaches are likely to show unsatisfactory performance due to the broad changes in the appearance of mitosis. What's more, the mimics often have a similar appearance and are wrongly identified as mitoses by these hand-crafted features. In a recent work [15], the task of mitosis detection was partly solved by deep convolutional neural networks (CNN), which are able to learn high-level feature representations and show superior detection results compared to other methods. This approach built a pixel-wise classifier using CNN, which is quite computationally intensive and time-consuming, which limits its practical clinical application.

To conquer the weaknesses of the above-mentioned approaches, [13] design an efficient and precise scheme to localize mitosis by developing a novel deep cascaded neural network (CasNN), which consists of two modules. First, a fully convolutional network (FCN)-based coarse retrieval model is proposed for identification and detection of mitosis candidates. This coarse retrieval module is able to find mitosis candidates within a whole microscopic image, and at the same time, keep a high sensitivity. Given those candidates, a fine discrimination module is designed for further discrimination of mitosis from similar mimics. Since the search range has been reduced from the whole image to only the candidates, the mitoses detection speed of [13] in a normal histology image is claimed to be 60 times faster than the state-of-the-art method [15]. To avoid overfitting caused by a limited number of training data and also boost the detection performance, the fine discrimination module is pre-trained on a large generalg image dataset, then fine-tuned on the mitosis detection dataset.

Figure 2.2 presents the schematic of approach [13], which is comprised of two cascaded convolutional neural networks working jointly. The model is a fully convolutional network, which is responsible for efficiently detecting mitosis candidates. It is referred to as the coarse retrieval model Nc, which produces a score map showing the probability of mitosis candidates existing in corresponding locations. Then the second model takes the detected

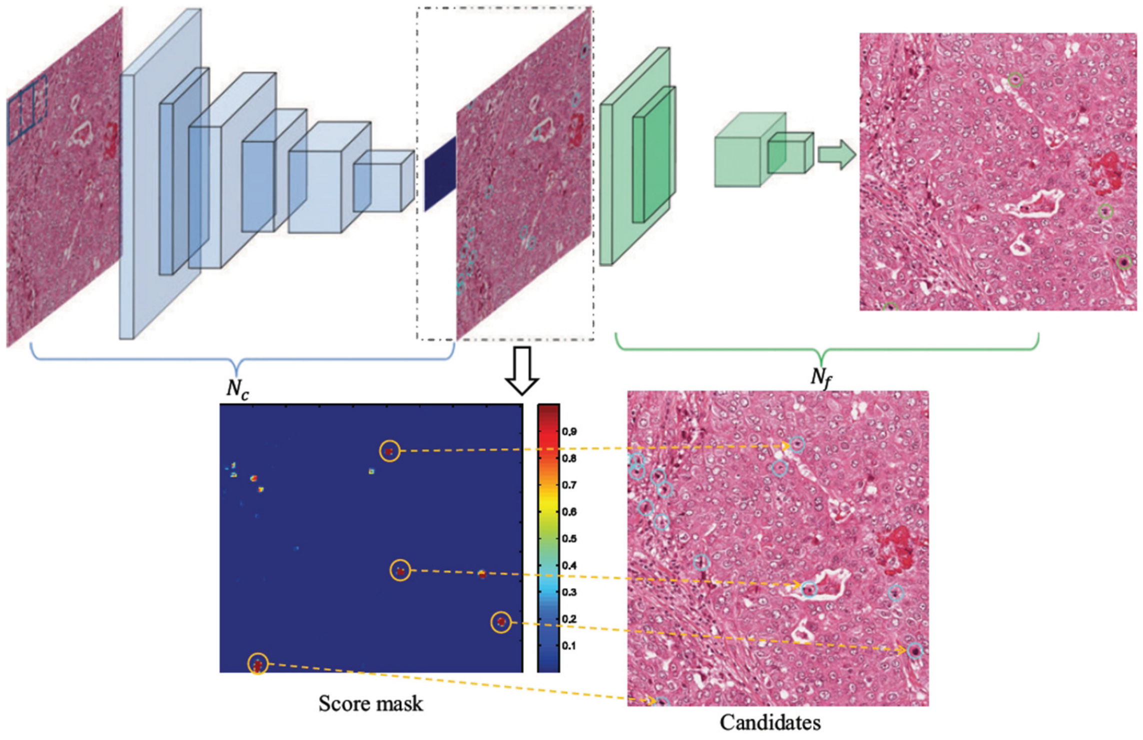


Figure 2.2: An overview of the deep cascaded networks for cell detection proposed in [13].

candidates as input for further recognition between mitoses and false targets with similar appearances. The second model is built by transferring deep and rich feature hierarchies trained by deep convolutional neural networks on a huge natural image dataset. The second is referred to as the fine discrimination model Nf, which has a stronger ability to build feature representations for input images than the CNNs that are barely trained on histopathological images, and is able to distinguish mitoses from hard mimics more accurately.

In recent years, several public cell detection contests have received considerable research attention, and also released their corresponding cell datasets, e.g., the MITOSIS contest at ICPR 2012 (Roux et al., 2013), the AMIDA13 contest at MICCAI 2013 (Veta et al., 2014), the MITOS-ATYPIA challenge at ICPR 2014, and the AMIDA16 contest at MICCAI 2016 with extensively enlarged data. These contests and datasets provide good opportunities for conducting experiments, algorithm evaluation, comparisons with cutting-edge methods, and are a significant contribution to the advances of cell detection research. In the experiments within this thesis, I also carried out considerable experiments on these challenging datasets.

Chapter 3

The Cell Counting Framework

3.1 Introduction

Cell counting is an important problem of counting or quantification of the number of cells for medical diagnosis and treatment, with several applications in research and clinical practice. For example, the complete blood cell count is able to assist a physician to determine the causes behind of patients' unwell feeling. Numerous sub researches in biology and medicine are in demand of the counting of cells. In medicine, the concentration of various blood cells, such as red blood cells and white blood cells, can give crucial information regarding the health conditions of a person. Studies that examine the growth speed of microorganisms (in other words: how fast they divide to create new cells) require cell counting. Many recent methods solve the problem of automated cell counting as an image analysis task, where they develop computer vision based recognition algorithms on high-quality microscopy images. A wide range of image analysis techniques can be employed for this purpose. In this chapter, we present a supervised learning framework with Convolutional Neural Network (CNN) and cast the cell counting task as a regression problem.

3.2 System overview

The system overview of our framework appears in Figure 3.1. In the training phase, a Convolutional Neural Network (CNN) is utilized to build a regression model between an image patch and its cell count number. We employ several kinds of CNN architectures and use Euclidean loss function during training, to enable the regression model to fit the cell counting task. To prepare the training data, we generate a large number of square patches from every training image. Along with each training patch, there is a patch count, which indicates the number of target cells present in the patch. After that, patch rotation is performed on the collected training patches for the purpose of making the system more robust to rotational variance and data augmentation.

In the test phase, one test image is cropped into a number of overlapping test patches with the same size as the training patches in the sliding-window manner. Each of these test

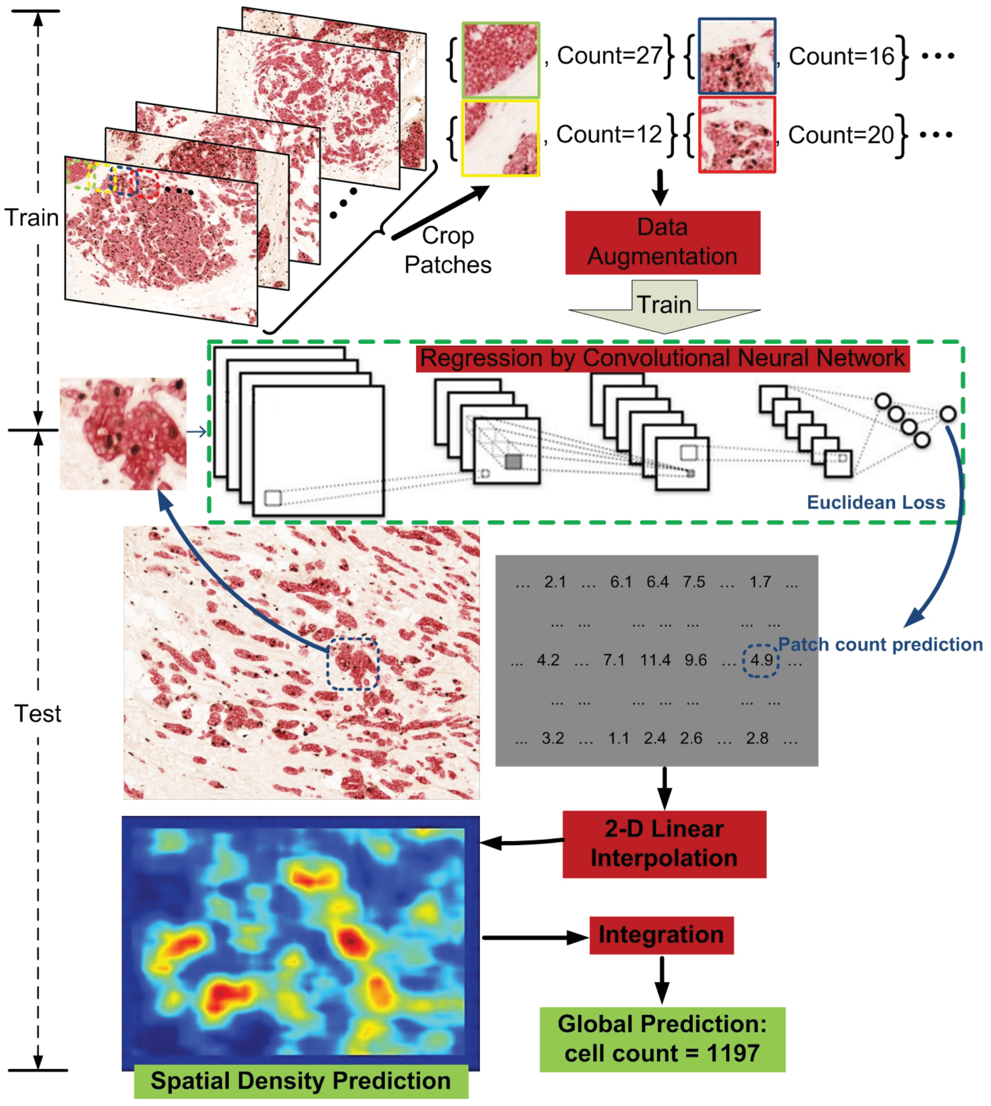


Figure 3.1: The system overview of the proposed cell counting framework.

patches is passed into the CNN-based regression model, and then the estimated cell count of the input test patch is output from the last layer of the CNN model. After predicting cell counts for all the patches, we perform a 2-D Linear Interpolation over the estimated cell count and its corresponding x-y coordinates to build a heatmap, which provides a spatial density prediction as shown in Figure 3.1. Lastly, integrating these interpolated counts on pixel locations provides us the final count on the test image. The whole procedure is illustrated in Figure 3.1.

3.3 Data Preparation and Processing

In automatic cell counting task, the resolution of a miscroscopy image can be high, at the same time the target cells could be very dense. Consequently, it is quite difficult to manually count target cells one-by-one. This is the principal motivation for deleoping an automatic cell counting method. Considering the nature of these medical images, which need automatic cell counting, data preparation and processing is naturally necessary. In this work, we crop an image into consistent patches and then perform training and prediction over these patches, in order to (1) make the approach more robust to scale variance, (2) avoid resizing original microscopy image, which could cause information loss, (3) prepare more training data to prevent the CNN based regression model from overfitting during training.

The proposed method operates by first partitioning an image into smaller patches. Patches are generated in a sliding window manner: from the top-left corner of a large W-by-H image with a certain patch size and stride size. Usually, stride size is set smaller than half of path size to ensure that adjacent patches have overlapping region. To construct training data, every training patch is accompanied by a patch count, which is an integer indicating how many cells exist in the patch. Then for data augmentation, a training patch is rotated from 0 degree to 360 degree with a certain rotation step, for example 30 degrees.

3.4 Convolutional Neural Network Regression Model

3.4.1 Classification vs Regression for Counting

As we know, in a CNN-based classification model, the network outputs a vector whose size is the same size as the number of classes. The i-element in the vector describes the confidence score that the input image belongs to the i-th class. During test phase, the index with the highest confidence score is selected as the final classification result. Softmax loss is widely used for classification problem.

However for counting problems, it is not proper to take cell count number as class index. The reason why regression is a better choice than classification for counting task has been explained in the background section of Chapter 2. In our counting-by-regression model, the difference between ground-truth value and the estimated value can be better preserved during calculating the error. This information is quite helpful for optimizing the CNN weights more accurately in the back-propagation phase. The layer of our regression model outputs a single number, indicating the number of cells that our model predicts. In our model, we employ two kinds of CNN architectures, the first one is AlexNet [52] which consists of 5 convolution layers + 3 fully connected layers; the other is the deep residual network (ResNet) [36]. In both of these architectures, the loss function is defined as the Euclidean loss, which measures the sum of squares of differences between the ground truth and prediction. We train the AlexNet model from scratch with Softmax and Euclidean loss layer respectively, the performance improvement of regression over classification is experimentally explained in Table 3.1. And Figure 3.4 provides 6 examples of cell counting result by the proposed counting-by-regression method.

3.4.2 Deep Residual Network for Regression

Since the 2012 ImageNet competition, convolutional neural networks have become popular in large scale image recognition tasks, several milestone networks (including AlexNet [52], VGGNet [44] and GoogLeNet [86], etc) have been proposed. Recently, the introduction of residual connections into deep convolutional networks has yielded state of the art performance in the 2015 ILSVRC challenge. This raises the question of whether there is any benefit in introducing deep residual network (ResNet) [36] into the cell counting task. In the following section, we are going to explain the network architecture and its components used in this chapter.

Convolutional layer. It consists of a set of learnable filters. During the forward pass, we slide each filter along the width and height of the input volume and compute dot products between its weights and the activation map from previous layer. Intuitively, the filters will be trained to be active to some type of visual feature such as an edge of some orientation or a blotch of some color on the first layer.

Pooling layer. It works by down-sampling the convolutional features using the max operation (*max-pooling*) or average operation (*average-pooling*). Pooling layer is usually inserted between successive convolutional layers, in order to reduce the amount of network parameters and also to control overfitting.

Batch Normalization layer. The ResNet [36] architecture uses Batch Normalization (BN) layer [39] right after each convolution and before activation. Normalization is often used as a pre-processing step to make the data consistent. When the input flows through a deep network, the weights and parameters adjust the values of the input, sometimes making the data too big or too small again. Batch Normalization layer allows us to normalize the data in each mini-batch across the network rather than just performing normalization once in the beginning, thus this problem is largely avoided. [39] has demonstrated that batch normalization helps to boost the learning speed and also increase the overall accuracy.

Fully-Connected layer. As the name implies, each neuron in a Fully-Connected (FC) layer has full connections to all neurons in the previous layer. After gathering all the responses from previous layers into each of its neuron, fully connected layer is responsible for computing a class-specific confidence vectors, where its each neuron outputs a score for a certain class. For example, the ResNet ends with a 1000-way fully-connected layer, on which the class with maximum score is selected as its final predicted label.

Overall Architecture.

Different from other CNN architectures, ResNet consists of a number of Residual Blocks. Each residual block is a made up of Convolutional layer, Batch Normalization layer and a shortcut that connects the original input with the output as shown in Figure 3.2 (a) and (b), where a Residual Block with Identity Shortcut (RB-IS) and a Residual Block with Projection Shortcut (RB-PS) is illustrated, respectively. The mathematical model of residual block can be summarized as:

$$y_l = F(X_l, \{W_l\}) + h(X_l) \qquad X_{l+1} = f(y_l) \tag{3.1}$$

$$h(X_l) = \begin{cases} X_l & \textit{identity mapping} \\ W_p X_l & \textit{projection mapping} \end{cases} \tag{3.2}$$

X_l and X_{l+1} are the input and output of the l-th residual block, $F(X_l, \{W_l\})$ stands for the residual function, and f is a activation function (e.g. ReLU). $h(X_l)$ represents the shortcut connection: identity mapping or projection mapping. If the dimension of X_l and X_{l+1} is the same, the identity shortcuts is used; otherwise a linear projection W_p is performed on the shortcut connections to match the dimension, that is projection mapping. The central idea [36] of ResNet is to learn the additive residual function F with respect to $h(X_l)$, with a key choice of using an identity mapping and/or projection mapping.

Figure 3.2 (a) and (b) show two types of Residual Blocks, which are used in different layers of ResNet model (c) according to whether the dimensions of input and output are the same. Nr_1, Nr_2, Nr_3 and Nr_4 represent the number of residual blocks used in four sections of ResNet model. For example, Nr_1=3, Nr_2=4, Nr_3=6 and Nr_4=3 in ResNet-50. Additionally, it has been demonstrated that pre-trained network can be adjusted to be effective for other computer vision tasks. We modify the last fully-connected layer of ResNet from outputting a 1000-D vector to outputting 1 item indicating the predicted number of cells. Additionally, we replace the softmax loss with Euclidean loss. After that, we perform fine-tuning on the weights in fully-connected layer of the ResNet using cell datasets, the parameters of previous layers are preserved. Finally, we obtain three ResNet based regression models for cell counting.

3.5 Datasets and Evaluation Metric

First, we describe the three cell datasets, on which the proposed method and other comparison methods are evaluated. The first dataset [45] involves 100 stained histology images

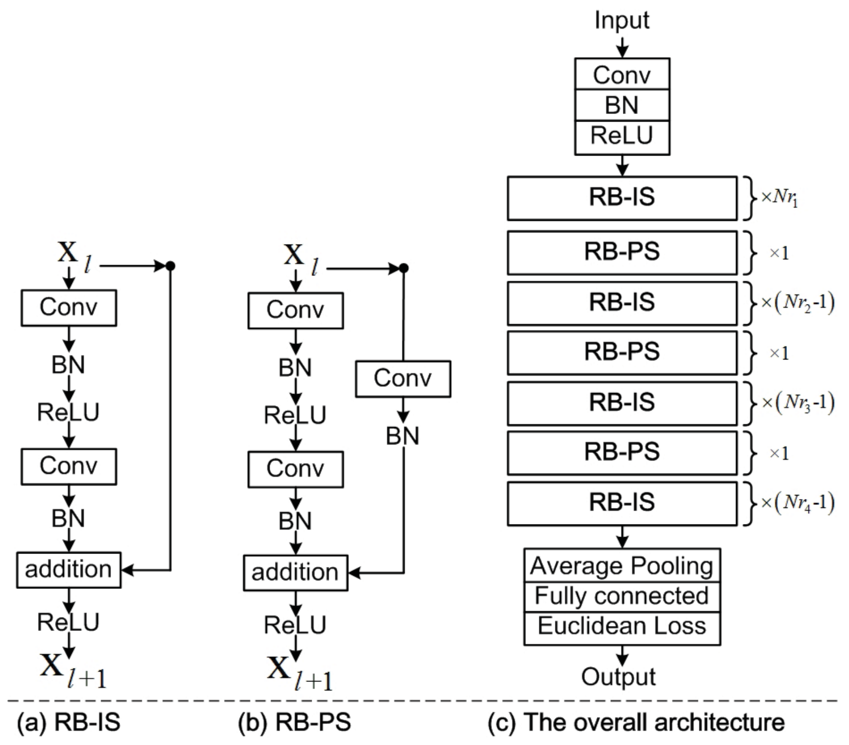


Figure 3.2: (a) RB-IS stands for the Residual Block with Identity Shortcut; (b) RB-PS is the Residual Block with Projection Shortcut; (c) An illustration of the architecture that we used in this chapter.

of colorectal adenocarcinomas. A total of 29,756 nuclei were marked at/around the center. The second dataset [57] consists of 200 highly realistic synthetic emulations of fluorescence microscopic images of bacterial cells. The third dataset [102] comprise of 55 high resolution RGB images, each of them is a microscopic image of proliferative tumor cells area with a high resolution of 1920-by-2560 pixels. The tumor cell diameter is about 10-20 pixels or 10 micrometer in physical length.

Table 3.1: *Size* is the image size; *Ntr/Nte* is the number of images selected for training and testing; *AC* indicates the average number of cells; *MinC-MaxC* is the minimum and maximum numbers of cells.

Cell Dataset	Size	Ntr/Nte	AC	MinC-MaxC
Nuclei [45]	500×500	50/50	310.22	1-1189
Bacterial [57]	256×256	100/100	171.47	74-317
Ki67 Cell [102]	1920×2560	45/10	2045.85	70-4808

To build this Ki-67 cell image dataset, a 10X microscopic field representing the highest proliferative area was acquired using a Nikon Eclipse E600 microscope with 0.25 aperture and a QImaging Micropublisher 5.0 RTV camera equipped with a Sony ICX282 CCD, finally it gives us 24-bit color pictures with a resolution of 1920 x 2560 pixels. All of the three evaluation cell datasets have their dotted annotation available, which represents the

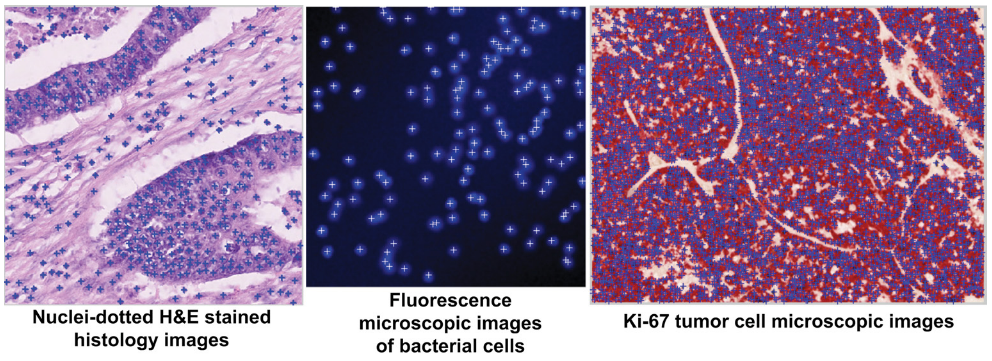

Nuclei-dotted H&E stained histology images
Fluorescence microscopic images of bacterial cells
Ki-67 tumor cell microscopic images

Figure 3.3: Example of the three evaluation datasets and dotted annotation. The three datasets are used in both cell counting and cell detection for evaluation.

location of cells as shown in Figure 3.3. For the three datasets, we randomly select images for training and testing. Details of the three evaluation datasets are summarized in Table 3.1.

In all the experiments, we use the Mean Relative Error (MRE) and Mean Absolute Error (MAE) as the metric for quantitative evaluation: where N is the total number of test images, t_i and p_i are the true and predicted numbers of cells in the i-th test image. MRE and MAE are defined as follows:

$$MRE = \frac{1}{N}\sum_{i=1}^{N}\frac{|t_i - p_i|}{t_i} \tag{3.3}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|t_i - p_i| \tag{3.4}$$

3.6 Implementation Details

The proposed method is implemented in Matlab, and we utilize Caffe [38], a fully open source implementation of Convolutional Neural Network, which affords clear access to Matlab/Python with support for GPU computation. As discussed in image partion part, each original RGB images is partitioned under certain rotation, stride size, and patch size. After taking experiments under different settings, we use stride size = 30 (pixels), patch size = 60×60 (pixels) and rotation step = 30 for the nuclei data; stride size = 20 (pixels), patch size = 40×40 (pixels) and rotation step = 30 for the bacterial data; stride size = 50 (pixels), patch size = 200×200 (pixels) and rotation step = 30 for the Ki-67 cell data. All the experiments are run on a machine with Intel Core i7-4790K CPU@4.00 GHz x 8 and GPU GeForce GTX TITAN Black/PCIe/SSE2.

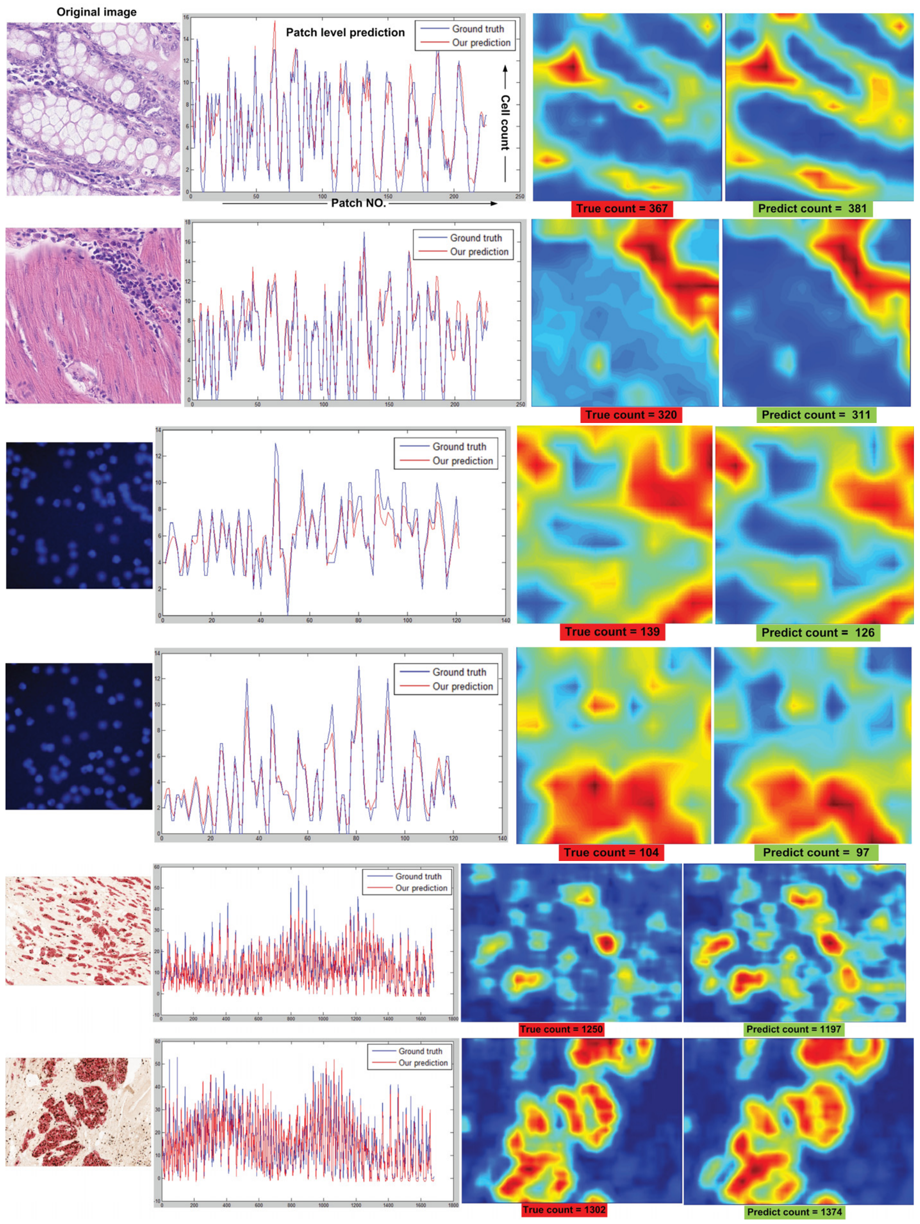


Figure 3.4: Spatial density prediction and counting results on the cell datasets. Original cell image is shown in on left side; the middle panel shows the patch level prediction, which is a middle result of our cell counting result; The right panel shows the spatial density prediction map (measured in number/square pixel) as well as the global counts of ground-truth and our prediction. From the patch level prediction curve, we can see that our estimated counts for each patch approximate the pattern of ground truth counts well.

3.7 Counting Performance using Different Models

First, we are going to investigate the performance difference between Classification (C) model and Regression (R) model for the cell counting task. The whole framework follows the pipeline shown in Figure 3.2. As for the CNN architecture, we employ AlexNet (5conv+3fc) and ResNet (50 layers) separately. And softmax loss function and Euclidean loss function are used respectively in the classification and regression model. We conduct this comparison experiment on all the three cell datasets and evaluate the performance in terms of Mean Relative Error and Mean Absolute Error (std also provided). Table 3.2 shows that on all the three datasets, the regression model gives lower prediction error by considerable margins than the classification model, which experimentally support the previous discussion. It is also necessary to note that the AlexNet based regerssion model outperforms the ResNet based classification model. Table 3.3 shows the counting performances using ResNets with different number of layers. The 50/101/152-layer ResNet based regression models are used in this experiment. We can observe that ResNet-152 model shows the lowest prediction error, followed by ResNet-101 and ResNet-50 respectively.

Table 3.2: Counting performance in terms of MRE and MAE±std comparison between Classification (C) and Regression (R) model

MRE	**Nuclei-dataset**	**Bacterial-dataset**	**Ki67-dataset**
AlexNet(C)	0.2175	0.0918	0.1226
AlexNet(R)	0.2019	0.0651	0.0959
ResNet(C)	0.2104	0.0772	0.1170
ResNet(R)	0.1925	0.0539	0.0775

MAE±std	**Nuclei-dataset**	**Bacterial-dataset**	**Ki67-dataset**
AlexNet(C)	20.7636±13.9416	12.9667±4.5361	213.5302±65.7220
AlexNet(R)	18.5720±12.6055	9.2591±3.3142	151.2059±44.6032
ResNet(C)	19.8742±13.5217	11.8711±3.8247	169.5076±50.2124
ResNet(R)	17.1437±11.5073	8.2064±2.8515	128.7426±40.5621

Table 3.3: Counting performance in terms of MRE and MAE±std using different models.

MRE	**Nuclei-dataset**	**Bacterial-dataset**	**Ki67-dataset**
ResNet-50(R)	0.1925	0.0539	0.0775
ResNet-101(R)	0.1845	0.0507	0.0697
ResNet-152(R)	0.1666	0.0450	0.0641

MAE±std	**Nuclei-dataset**	**Bacterial-dataset**	**Ki67-dataset**
ResNet-50(R)	17.1437±11.5073	8.2064±2.8515	128.7426±40.5621
ResNet-101(R)	16.3164±10.8762	7.7542±2.4580	116.3076±39.0215
ResNet-152(R)	14.9275±10.4368	7.4741±2.2248	108.3014±40.4698

3.8 Comparison with state of the art

We carry out experimental performance comparison between our method and three other state of the art approaches (presented in [6], [57], [79]) on three evaluation datasets. The counting result from ResNet-152 regression model is used in our approach. Figure 3.5 provides the cell counts of ground-truth and four predictions of "Le.count" [57], "Le.detect" [6], "DeepFeat" [79] and "The proposed" on every test image. To quantify Figure 3.5, Table.3.4 reports the performance in terms of Mean Relative Error (MRE) and Mean Absolute Error (MAE) over the three evaluation datasets.

The proposed method has achieved very competitive result on Nuclei dataset and Ki67 dataset, MRE=16.66% and 6.41% respectively. The images from Nuclei and Ki67 datasets contain 310.22 and 2045.85 cells on average, our proposed method is able to predict with only 14.93 and 108.30 cells in terms of mean absolute error; while other three methods gives 33.89-71.80 and 189.35-259.67 error cells on average.

On Bacterial dataset, the proposed method gives MRE=4.50%, but [101] makes 2.4% further improvement over our result. The central idea of [101] is to estimate a density function whose integral over any image region gives the count of objects within that region. In its learning phase, each cell is dot-annotated and is assigned a real-valued Sift feature vector describing the local appearance. It means that for each cell, [101] needs its x-y coordinate on an image and then compute the Sift feature on the image sub-region around this cell. In comparison, the proposed method only takes the number of cells as annotation to an image patch during training. As one can imagine, for an image containing hundreds to thousands of cells, the complexity and time consuming of [101] will increase greatly. Furthermore, when it comes to the much more dense datasets (Nuclei and Ki67), the Sift descriptor based learning of [101] becomes less reliable.

It is also necessary to mention that the MRE values of all the four methods on Nuclei dataset are higher than those on other two datasets, because Nuclei dataset has several test images, which only contains a few cells e.g. 1, 4 or 8. For example, predicting the cell count from 1 (ground truth) to 2 or from 4 (ground truth) to 6 will greatly affect the final MRE value.

3.8.1 Comparison with FCN based Cell Counting Scheme

Recently, fully convolutional network (FCN) has been applied to the cell counting task by [101], where FCN predicts a spatial density map of target cells. Then, the number of cells can be estimated by integration over the density map, as in the proposed framework. We follow this FCN-based method and carry out comparison experiment on the evaluation datasets. We have observed that the density map predicted by FCN is not accurate for cell counting task in several cases. Figure 3.6 illustrates the density map predicted by FCN and our method, where the estimated cell count is also shown in the figure caption. As shown in the Figure 3.6, FCN generates a cell density map, however for some images, where target

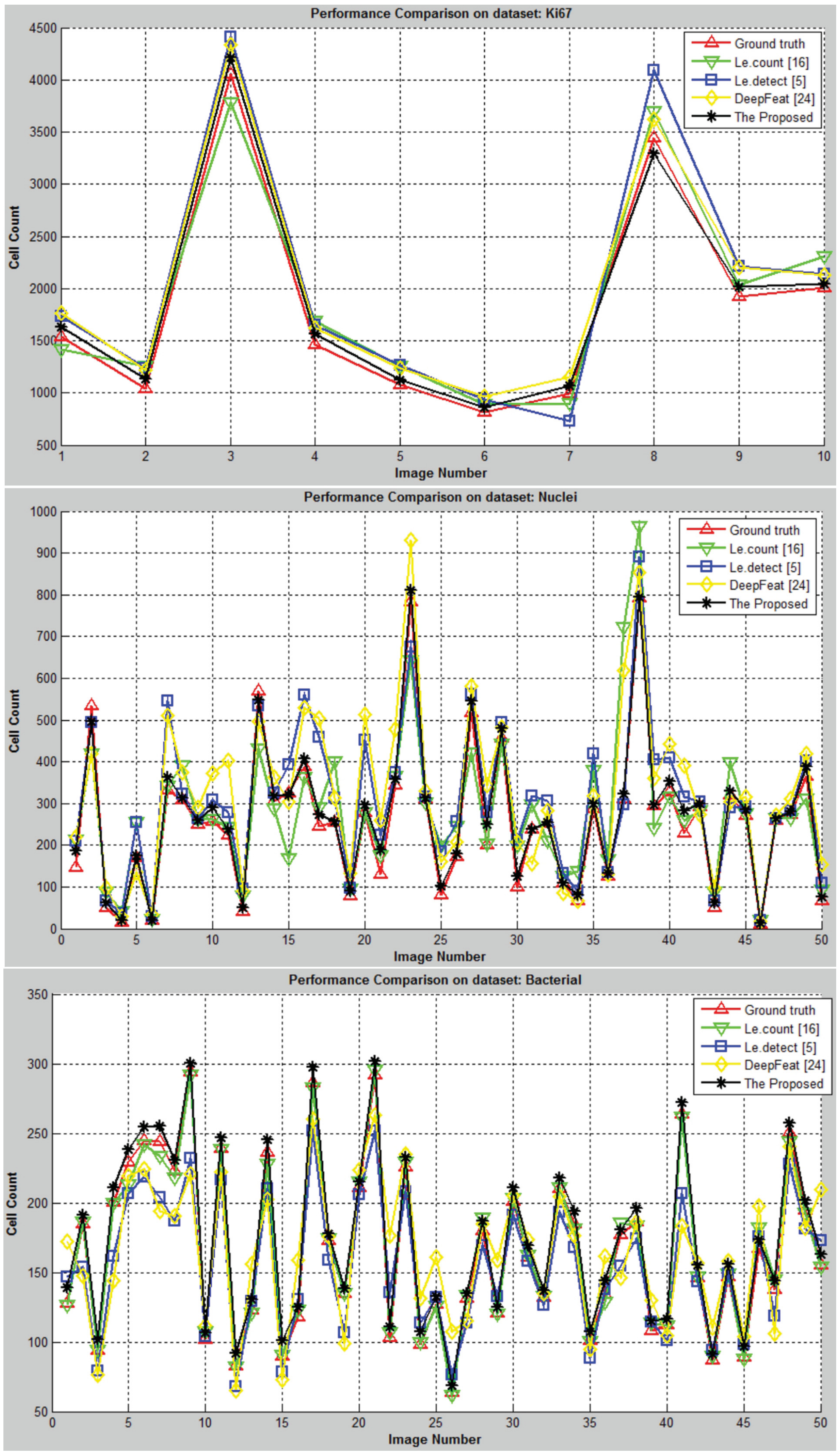


Figure 3.5: The estimated count versus ground-truth of different approaches on the three evaluation datasets.

Table 3.4: Counting performance (MRE) and (MAE±std) comparison on the three evaluation datasets.

MRE	**Nuclei-dataset**	**Bacterial-dataset**	**Ki67-dataset**
DeepFeat [79]	0.3581	0.1751	0.1249
Le.count [57]	0.2674	**0.0208**	0.1151
Le.detect [6]	0.3206	0.1083	0.1540
The proposed	**0.1666**	0.0450	**0.0641**

MAE±std	**Nuclei-dataset**	**Bacterial-dataset**	**Ki67-dataset**
DeepFeat [79]	71.8046 ± 51.4109	25.4792 ± 19.1504	189.3559 ± 53.6329
Le.count [57]	51.4479 ± 39.8087	**6.4061 ± 3.5657**	185.9391 ± 60.5042
Le.detect [6]	33.8995 ± 23.9252	18.1937 ± 13.4393	259.6736 ± 85.0594
The proposed	**14.9275 ± 10.4368**	7.4741 ± 2.2248	**108.3014 ± 40.4698**

cells are extremely dense just like the 3th and 4th examples, the cell density map is not accurate enough compared with our prediction. And for most cases, those strong activation regions are blurry (like 1th and 2th examples) or merged with neighboring cells (like 3th and 4th examples). Consequently, the total cell counts estimated by FCN is less reliable compared with our estimated cell counts, especially in cell-dense images.

3.9 Challenge of Imbalanced Data

Another significant challenge is the imbalanced data that is also a common problem in many medical image analysis tasks and can be characterized as having many more instances of certain classes than others. Our observations are as follows:

1. As we know, a machine learning model (e.g. CNN) is trained by iteratively adjusting the weights, so that the error between the computed output of the network and the targeted output is minimized. During training, the error (i.e. cell count error) of sparse images (i.e. images containing a few cells) is much smaller than the counting error for dense images (i.e. images containing a lot of cells). Therefore, it has been observed that the gradient vector will be dominated by that of the dense images. Consequently, if weights are recalculated in the direction of the gradient vector, the network error of the dense images will decrease significantly and that of the sparse images may even increase significantly.

2. As a consequence, sparse images will suffer more from the imbalanced data problem compared to the dense images. This is what we observe in Figure 3.7, that the prediction error rates of images with sparse-disease-cells is higher than images with dense-disease cells, in general.

3. Furthermore, in the real-world application of cell counting, most test images have sparse or medium number of target cells, images with dense cells are rare. Thus, the imbalanced data problem has become a major difficulty to the cell counting task. Unfortunately,

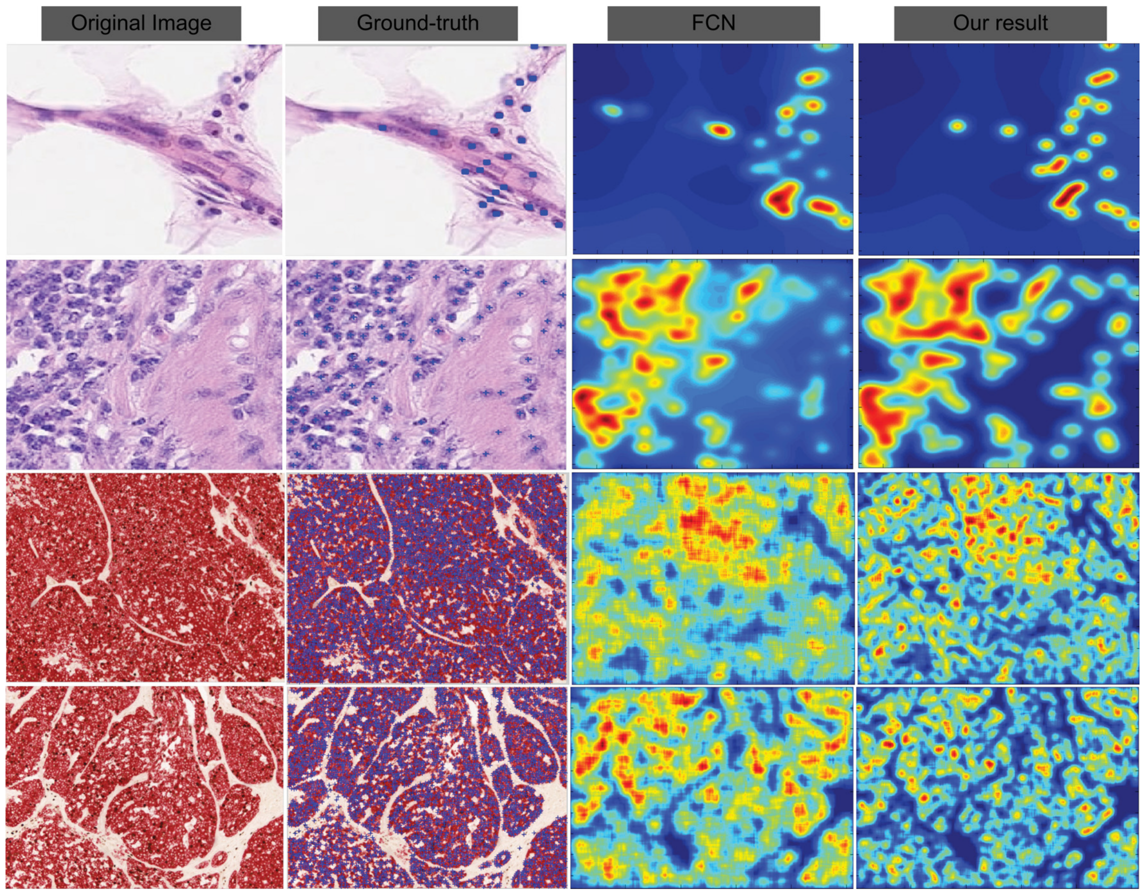


Figure 3.6: Each row corresponds to an input example and its prediction results by two approaches. The cell counts of ground-truth, FCN's result and our result are respectively, first row: $\{24, 27.13, 25.60\}$, second row: $\{112, 120.54, 117.26\}$, third row: $\{4157, 4372.49, 4239.30\}$, fourth row: $\{3487, 3621.33, 3408.14\}$.

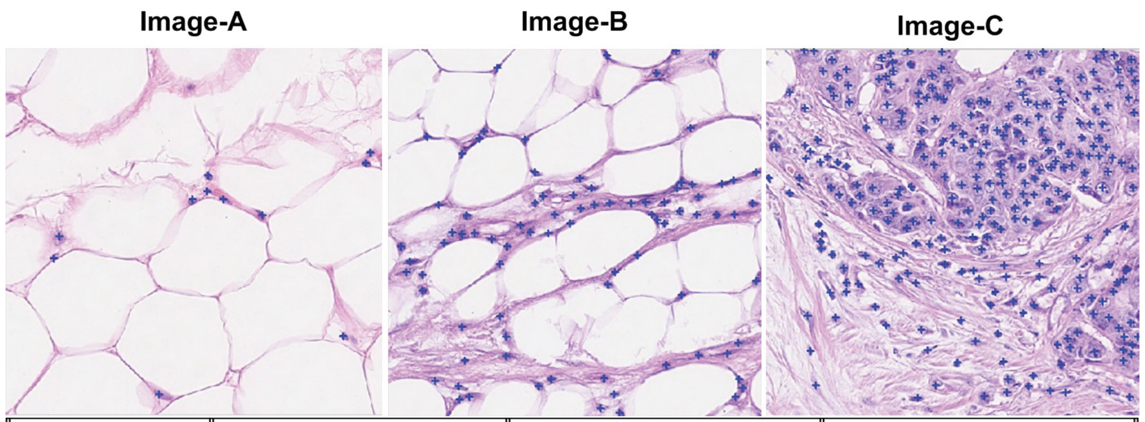


	Image-A, T=11		Image-B, T=57		Image-C, T=278	
FCN	P=5.4	E=50.91%	P=62.9	E=10.35%	P=268.5	E=3.42%
Proposed-B	P=9.1	E=17.27%	P=61.2	E=7.37%	P=270.2	E=2.81%

Figure 3.7: Three cell images with sparse/medium/dense target cells. T, P and E indicates the True cell count, Predicted cell count and Error rate. The first row shows the estimation result of FCN-based counting method, which is the-state-of-art approach recently. The second row is the estimation result of the proposed approach with data balancing. It is observed that the prediction error rates of images with sparse-disease-cells is higher than those of images with dense-disease-cells.

most existing cell counting approaches assume a relatively balanced data distribution effectively ignoring the imbalanced data issue.

3.9.1 Balancing Training Data

Considering the nature of these medical images, data preparation and pre-processing is necessary. In this work, we first crop the training images into consistent patches and then perform training and prediction over these patches, in order to (1) make the approach more robust to scale variance, (2) avoid resizing original microscope image, which could cause information loss, (3) prepare more training data to prevent the CNN based regression model from overfitting during training.

The proposed method operates by first partitioning images to smaller patches. Patches are generated with a certain patch size and stride size. Usually, stride size is set smaller than half of patch size to ensure that adjacent patches have overlapping region. To construct training data, every training patch is accompanied by a patch count, which is an integer indicating how many cells exist in the patch.

For data augmentation, a training patch is rotated from 0 degree to 360 degree with a variable rotation step. We change this rotation step with respect to cell density per patch, in order to make our training data more balanced. The details for balancing training data is discussed in the following paragraphs.

As discussed in introduction part, a machine learning model (e.g. CNN) is trained to

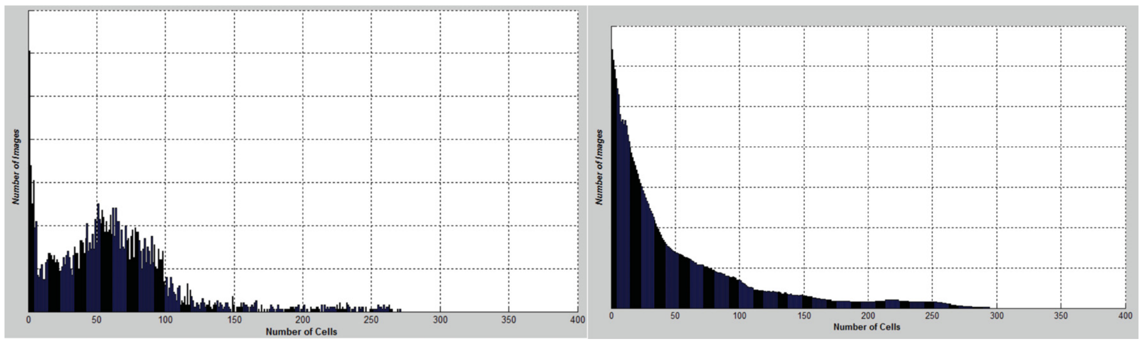


Figure 3.8: Left panel shows the original distribution of a training data. Right panel shows balanced data distribution after augmenting number of sparesely populated patches by rotations.

produce a smaller overall loss or error during the training phase. However, due to the presence of imbalanced training data, the error of the dense images will decrease significantly and that of the sparse images will increase significantly, typically. Thus, during prediction cell counts for densely populated images will be more accurate than for the sparsely populated ones.

Given the above observation, we are motivated to tune the distribution of training data to be more balanced by data augmentation. It means that for sparse images, we perform patch rotation with a smaller rotation step, so that more training images are available for sparse side, conversely less training images for dense side. Compared to building a weighted loss function and modifying corresponding back propagation of neural network, the proposed scheme is a convenient solution to the imbalanced data problem, and can be easily applied to other existing cell counting approaches. After making the training data more balanced with respect to cell density as shown in the Figure 3.8, we train the proposed approach again on the balanced data, which is indicated as "Proposed-balanced." Note that shape of the balanced data histogram is (roughly) inversely proportional to the number of cells.

3.10 Experiment after Balancing Cell Data

Here one further experiment is made to explore the influence of imbalanced data. We organize this experiment in the following steps:

1. We generate about 2500 test patches in size of 224-by-224, and the number of cells in those patches are in the range of [0,273], which covers the cell density range from "very sparse" to "very dense". (Actually, more than 130 cells in a 224-by-224 patch is already quite dense.)

2. We run FCN-based counting and our counting approaches on the 2500 test patches to collect their estimated cell counts. The counting results estimated by FCN, the proposed approach and the proposed approach with balancing data are indicated as "FCN" "Proposed" and "Proposed-balanced" in the following part.

3. To visualize the result, we rank ground-truth cell counts in the ascending order, so

that the x-axis in Figure 3.9 represents the cell density from "sparse" to "dense". Then we compare the cell counts given by three approaches under an environment, where the cell density increases (as shown in the top-left image in Figure 3.9). Furthermore, we compute the "Error=Estimate-True" and "(Estimate-True)/True" for better visualization and analysis (as shown in the top-right and bottom-right image in Figure 3.9).

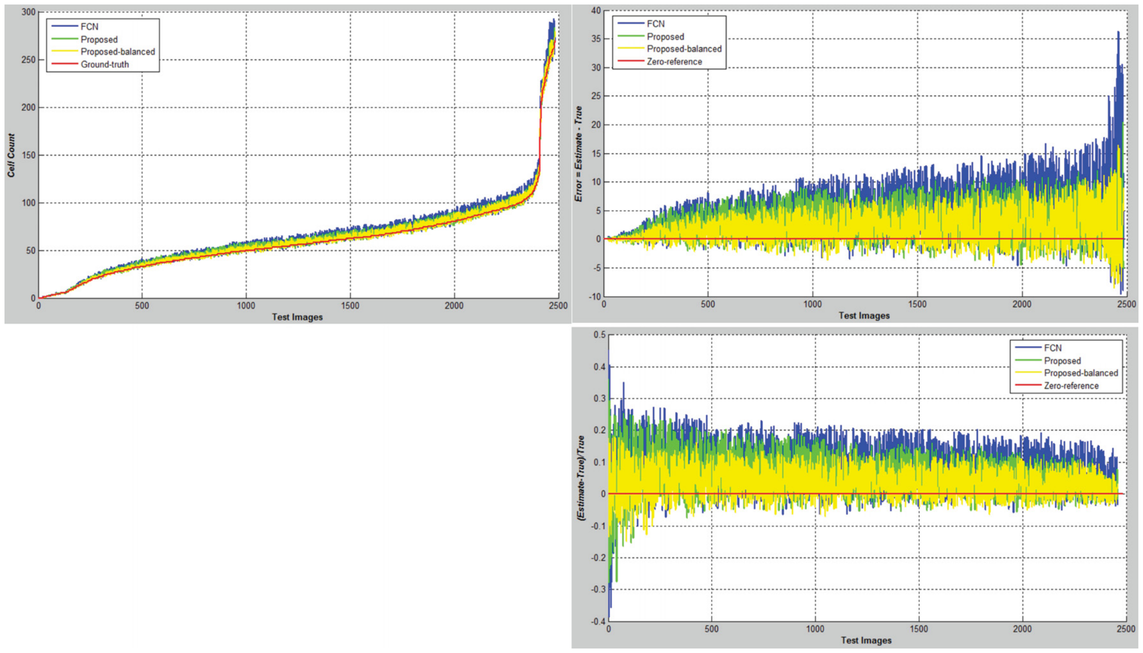


Figure 3.9: Accuracies of FCN-based and proposed cell counting approaches with increasing cell density.

From the top-left panel of Figure 3.9, we observe that all the three methods (FCN, Proposed, Proposed-balanced) try to approximate the ground-truth curve. From the bottom-right of Figure 3.9, the performance difference between the three methods becomes clear. It shows that with the increasing of cell density, the relative error of FCN increases much more quickly than those of Proposed and Proposed-balanced. We believe the issue of data balancing is behind such behavior. Note that in an image the density map varies from one part to another and the density map also varies across training images. Because, working with Euclidean loss, FCN does not compensate for this data imbalance. As a result, it produces large relative counting errors for sparely populated areas or images. Another practical disadvantage of FCN that its prediction is dependent on smoothing kernels used over dotted annotations - it is difficult to use a single smoothing kernel that works over a range of densely and sparsely populated cell images.

For the proposed method, we observe that when we compensate for the data imbalance, we achieve better accuracy. In dense patches side, Proposed-balanced performs slightly better than the Proposed; in the sparse patches side, Proposed-balanced performs much better than Proposed and FCN.

3.11 Performance and Comparison with State-of-the-art

We carry out experimental accuracy comparison between our method and 4 other state of the art approaches: “Le.count” [57], “Le.detect” [6], “DeepFeat” [79] and “FCN.count” [101]. “Proposed” and “Proposed-Balanced” refer to our method without and with data balancing, respectively. Table 3.5 reports accuracies in terms of Mean Relative Error (MRE) and Mean Absolute Error (MAE) for the three evaluation datasets.

Table 3.5: Counting performance (MRE) and (MAE±std) comparison on the three evaluation datasets.

MRE	**Nuclei-data**	**Bacterial-data**	**Ki67-data**
DeepFeat [79]	0.3581	0.1751	0.1249
Le.count [57]	0.2674	**0.0208**	0.1151
Le.detect [6]	0.3206	0.1083	0.1540
FCN.count [101]	0.2514	0.0843	0.1087
Proposed	0.1666	0.0450	0.0641
Proposed-Balanced	**0.1436**	0.0338	**0.0527**

MAE±std	**Nuclei-data**	**Bacterial-data**	**Ki67-data**
DeepFeat [79]	71.80 ± 51.41	25.47 ± 19.15	189.35 ± 53.63
Le.count [57]	51.44 ± 39.80	**6.40** ± 3.56	185.93 ± 60.50
Le.detect [6]	33.89 ± 23.92	18.19 ± 13.43	259.67 ± 85.05
FCN.count [101]	25.25 ± 17.75	14.54 ± 7.65	134.87 ± 46.55
Proposed	14.92 ± 10.43	7.47 ± 2.22	108.30 ± 40.46
Proposed-Balanced	**13.54 ± 8.76**	6.79 ± **2.07**	**105.81 ± 37.63**

With data balancing the proposed method has achieved very competitive result on Nuclei dataset and Ki67 dataset, MRE=14.36% and 5.27% respectively. The images from Nuclei and Ki67 datasets contain 310.22 and 2045.85 cells on average, our proposed balanced method is able to predict with only 13.54 and 105.81 cells in terms of mean absolute error; while other three methods gives 33.89-71.80 and 189.35-259.67 error cells on average.

3.12 Summary of this chapter

In this chapter, we present a novel regression based method for cell counting using latest architectures of convolutional neural network. As the output, spatial density map and global cell count are provided. The proposed method is able to handle both dense and sparse cell microscopy images. We have experimentally demonstrated that the proposed approach achieved superior performance compared with several recent related methods.

Chapter 4

The Cell Detection Framework

4.1 Introduction

Output encoding often leads to superior accuracies in various machine learning tasks. In this chapter we look at a significant task of cell detection/localization from microscopy images as a test case for output encoding and supervised learning with a convolutional neural network (CNN). Since the output space is sparse for the cell detection problem (only a few pixel locations are cell centers), we employ compressed sensing (CS)-based output encoding here. Using random projections, CS converts the sparse, output pixel space into dense and short (i.e., compressed) vectors. As a regressor, we use a deep CNN to predict the compressed vectors. Then applying a L_1-norm recovery algorithm to the predicted vectors, we recover sparse cell locations in the output pixel space. We demonstrate CS-based output encoding provides us with the opportunity to do ensemble averaging to boost detection/localization scores. We experimentally demonstrate that the proposed CNN + CS (referred to as CNNCS) framework is competitive or better than the the state of the art methods on benchmark datasets for microscopy cell detection. In the AMIDA13 MICCAI grand competition, we achieve the 3rd highest F1-score in all the 17 participated teams.

Output encoding transforms the labels (such as 1-hot vectors in classification) of training examples into a different representation, where an inverse transformation can be applied to recover the original label. Generally, an ensemble of machine learners is trained to predict the transformed label. Finally, an inverse transformation or decoding is applied to retrieve the original output labels.

One of the earliest research in the output encoding with error correcting ability [21] had shown superior accuracy. In the recent past, redundancy in the output representation [92] yielded more accurate predictions. The RAKEL method constructs many random a k-labelsets that are subsets of the multiple labels and trains classifiers for those. Then, it combines the ensemble by voting [92].

When the output label is sparse, a natural question for output encoding is how to exploit this sparsity. Eventually, output encoding of sparse vectors borrowed an elegant tool called compressed sensing or compressive sensing (CS) [23, 25] from the signal processing community. CS is theoretically and algorithmically rich and has several practical applications, such as reducing MRI scan time for patients and building a smaller and cheaper camera. Under the premise of CS, an unknown signal of interest is observed (sensed) through a limited number of linear observations. Then, it is possible to obtain a stable reconstruction of the unknown signal from these observations, under a general assumption that the signal is sparse or can be represented sparsely with respect to a linear basis [23, 25]. The signal recovery techniques typically rely on convex optimization with a L_1 norm regularization. Examples include orthogonal matching pursuit (OMP) [9], dual augmented Lagrangian (DAL) method [90], Learned Iterative Shrinkage-Thresholding Algorithm (LISTA) [33], etc.

The principle behind CS-based output encoding is straightforward. First, the (sparse) output label signal is projected to a shorter and dense vector. A machine learner is then trained to regress this short and compressed vector. A recovery algorithm, which is typically a L_1-norm convex optimization, recovers the sparse output vector from the predicted compressed vector.

In the past, CS-based encoding was used in conjunction with linear and non-linear predictions [37, 43, 47]. Hsu et al. [37] proves a generalization error bound for CS-based output encoding. Not surprisingly, the generalization error is bound by the sum of two components- the prediction error from the machine learner (regressor) and the reconstruction error of the recovery algorithm [37]. In particular, Hsu et al. [37] showed that theoretically, using a linear predictor along with CS-based encoding is no more difficult than using a linear predictor alone [37].

Use of non-linear predictors with CS-based output encoding is somewhat recent in machine learning. Viswanathan et al. [47] used Bayesian inference with CS and showed good accuracy in prediction. Recently, decision trees and gradient boosting have been used in conjunction with CS encoding to yield good prediction accuracies [43].

Continuing this trend, in this chapter, we combine CS-based output encoding with deep convolutional neural net (CNN). We refer to our proposed framework as CNNCS (convolutional neural network + compressed sensing). There are some advantages of using CS-based output encoding. First, the compressed output vector is much shorter in length than the original sparse pixel space. So, the memory requirement would be typically smaller and consequently, over-fitting in the CNN would be under control. Thus, compressing the output space can be viewed as a form of regularization on the network. Next, there are plenty of opportunities to apply ensemble average to improve generalization accuracy. In the present chapter, we exploit this opportunity by creating compressed but redundant representations. Furthermore, CS-theory dictates that pairwise distances in the sparse space are approximately maintained in the compressed space. So, even after the output space encoding, CNN still targets the original output space in an equivalent distance norm.

A significant test application for CNNCS is automatic cell detection from microscopy images. Automatic cell detection is to find whether there are certain types of cells present in an input image and to localize these cells in the image. It is of significant interest to a wide range of medical imaging tasks and clinical applications, such as diagnosis of breast cancer. To see, why the cell detection application fits the sparsity assumption in CS, consider the following. If there are 5000 cells present in an image of size 2000-by-2000 pixels, this fraction is $5000/(2000 * 2000) = 0.00125$, signifying that even a dense cell image is still quite sparse in the pixel space.

Our contributions in this chapter are as follows. First, this is one of the first attempt to combine deep learning with CS-based output encoding to solve cell detection and localization. Second, we introduce redundancies in the CS-based output encoding that are exploited by CNN to boost accuracy in cell detection and localization. Third, on benchmark datasets CNNCS achieves excellent accuracy compared to the state of the art method. In one such dataset, CNNCS secures the first position and in another it ranks second among its competitors.

4.2 Compressed Sensing Theory

During the past decade, compressed sensing or compressive sensing (CS) [23] has emerged as a new framework for signal acquisition and reconstruction, and has received growing attention, mainly motivated by the rich theoretical and experimental results shown in [67], [26], [23] and so on. As we know, the Nyquist-Shannon sampling theorem states that a certain minimum sampling rate is required in order to reconstruct a band-limited signal. However, CS enables a potentially large reduction in the sampling and computation costs for sensing/reconstructing signals that are sparse or have a sparse representation under some transforms (e.g. Fourier transform).

Under the premise of CS, an unknown signal of interest is observed (sensed) through a limited number of linear observations. It is possible to obtain a stable reconstruction of the unknown signal from these observations, under a general assumption that the signal is sparse or can be represented sparsely with respect to a linear basis [67], [26], [23]. The signal recovery techniques typically rely on convex optimization with a penalty expressed by L_1 norm, for example, the Orthogonal Matching Pursuit (OMP) [9] is a widely-used algorithm, which requires the degree of signal sparsity (i.e. the number of targets in detection problem) as a given variable. Dual augmented Lagrangian (DAL) method [90] is another choice, which doesn't need the signal sparsity degree as input. [33] proposed two versions of a very fast and trainable algorithm that produces approximate estimates of the sparse code that can be used to compute good visual features, or to initialize exact iterative algorithms. Its main idea is to train a non-linear, feed-forward predictor with a specific architecture and a fixed depth to produce the best possible approximation of the sparse code.

Three major components of CS theory can be generally summarized as: (1) to obtain observation vectors of the original sparse signal, (2) signal transmission or processing, (3) to recovery the original signal from the received observation vectors, which probably contains error or noise.

Suppose we have a n-dimensional vector (signal) a, if a is a sparse signal or could be represented sparsely in a certain transform domain, the CS theory guarantees that a can be recovered exactly by taking random measurements much less than n. If there are at most k non-zero entries in a, we say that a is k-sparse. In order to take the measurements, we first introduce the sensing matrix D, which is a $m \times n$ matrix with $m \ll n$, then the random non-adaptive measurements are obtained by a linear system:

$$x = Da \tag{4.1}$$

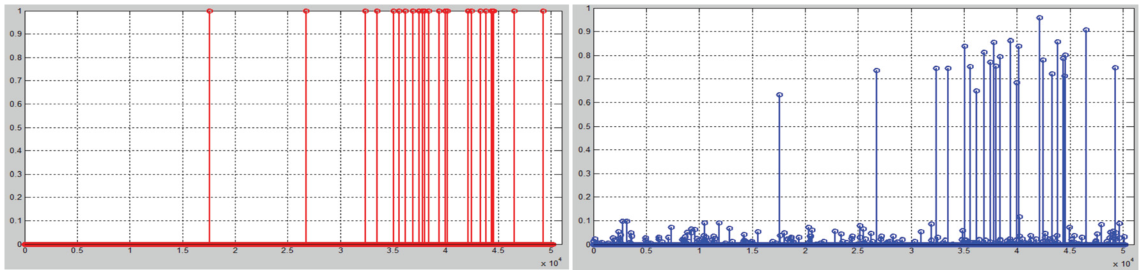


Figure 4.1: Original signal (red) and Recovered signal (blue) by L_1 minimization recovery.

The CS theory says that the signal can be recovered exactly if the number of measurements obeys the condition $m \geq (C_m)(k)log(n)$, where C_m is a small constant greater than one. The signal can be reconstructed by solving the following convex optimization problem:

$$\hat{a} = \arg\min_{a} \|a\|_1 \qquad \text{subject to} \qquad x = Da \tag{4.2}$$

The sensing matrix D should satisfy with Restricted Isometry Property (RIP) condition, which states that:

$$(1-\delta)\|a\|_2 \leq \|Da\|_2 \leq (1+\delta)\|a\|_2 \tag{4.3}$$

The restricted isometry property for any k-sparse vector. The restricted isometry constant is δ, $0 < \delta < 1$. RIP property implies that the sensing matrix D is guaranteed to only change the length of any vector a “very little” as long as the vector a is at least k-sparse. It has been proved that several matrices satisfy the sufficient RIP condition, like random Gaussian and partial Fourier matrices.

The central idea of compressed sensing (CS) theory [23] can be summarized that if a signal is sparse, then under certain sufficient condition, it can be reconstructed exactly from a small set of random linear measurements using tractable optimization algorithms. Figure 4.1 shows an example sparse signal and its recovery using CS techniques.

4.3 Compressed Sensing in Machine Learning

Literature is not abundant when it comes to combining CS and DL. DL architectures were applied for video compressive sensing [59]. A large improvement in reconstruction quality was obtained compared to existing approaches. In their work, a fully-connected neural network is trained to map directly temporal CS measurements to video frames. [38] developed a general theory for a variant of the popular error correcting output code scheme, using ideas from compressed sensing. They consider multi-label prediction problems with large output spaces under the assumption of output sparsity. The method can be regarded as a simple reduction from multi-label regression problems to binary regression problems. They also prove robustness guarantees for this method in the form of regret transform bounds (in general).

4.4 Compressed Sensing for Cell Detection

Suppose, we have a n-length signal a, which carries the location information of target cells. Because a is sparse, the CS theory guarantees that a can be recovered from linear observations x using (4.2).

As shown in Figure 4.2, for a training image, each cell location is converted to a positive element in the binary signal a. So, each positive element of a samples one column of the sensing matrix. This way the annotation mask B is encoded into the observed signal x. After $\hat{a}$ is recovered, it is easy to get the pixel-level annotations B. Since, $\hat{a}$ is equivalent to pixel-level annotations B, by applying a threshold (T) on $\hat{a}$ and reshaping back to B.

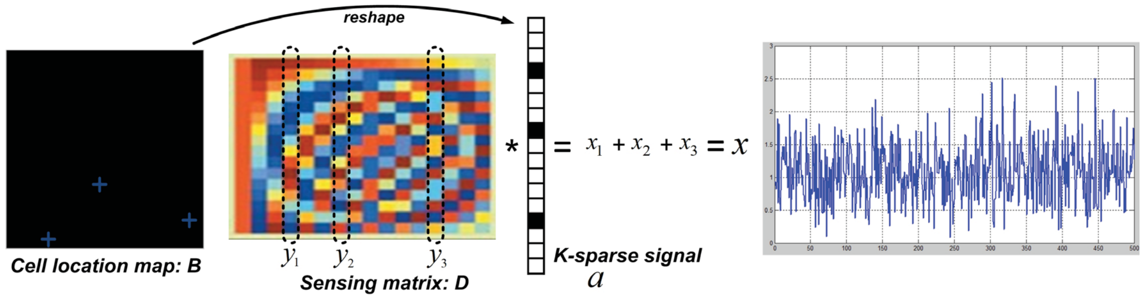


Figure 4.2: Converting a binary pixel-wise annotation to n-length real-value signal according to Compressed Sensing theory.

During the encoding phase, each pixel-level annotation is converted to a positive element in the binary signal a, then each positive element samples one column of the sensing matrix, finally the annotation mask B is encoded into a summation of these samples. It means that we build a n-length signal x as the representation of pixel-level annotations B.

Previously, it is the coordinates of annotations that carry the location information of a cell, consequently, small error or bias in B can result in a false detection. Now, the location information of cells is encoded into the pattern of the n-length signal x. It has been experimentally observed that accurate recovery of pixel-level annotations B can still be obtained from signal $\hat{x}$, even though every element of signal $\hat{x}$ has been disturbed by system error or bias to some extent. The whole process is shown in Figure 4.2.

4.5 System Overview

The proposed detection framework consists of three major components: (1) cell location encoding phase using random projection, (2) a CNN based regression model to capture the relationship between a cell microscopy image and the encoded signal x, and (3) decoding phase for recovery and detection. The flow chart of the whole framework is shown in Figure 4.3.

During training, the ground truth location of cells are indicated by a pixel-wise binary annotation map B. We propose two cell location encoding schemes, which convert cell location from the pixel space representation B to a compressed signal representation x. Then, training pairs, each consisting of a cell microscopy image and the compressed signal x, train a CNN to work as a multi-label regression model. We employ the Euclidean loss function during training, because it is often more suitable for a regression task. Image rotations may be performed on the training sets for the purpose of data augmentation as well as making the system more robust to rotations.

During testing, the trained network outputs an estimated signal $\hat{x}$ for each test image. After that, a decoding scheme is designed to estimate the ground truth cell location by performing L_1 minimization recovery on the estimated signal $\hat{x}$, with the known sensing matrix.

4.6 Cell Location Encoding and Decoding Scheme

4.6.1 Encoding Schemes

In the CNNCS framework, we employ two types of random projection-based encodings as described below.

Scheme-1: Encoding by Reshaping

For the cell detection problem, cells are often annotated by pixel-level labels. The most common way is to attach a dot or cross at the center of every cell, instead of a bounding box around the cell. So, let us suppose there is a pixel-wise binary annotation map B of size w-by-h, which indicates the location of cells by labeling 1 at the pixels of cell centroids, otherwise labeling 0 at background pixels. To vectorize the annotation map B, the most intuitive scheme is to concatenate every row of B into a binary vector a of length $w \times h$. Thus, a positive element in B with $\{i, j\}$ coordinates will be encoded to the $[i + h(j-1)]$-th position in a. a is also a k-sparse signal, so, there are at most k non-zero entries in a. Here, we refer this intuitive encoding scheme as "Scheme-1: Encoding by Reshaping".

After the vector a is generated, we apply a random projection. CS theory guarantees that a can be fully represented by linear observations x:

$$x = Da, \tag{4.4}$$

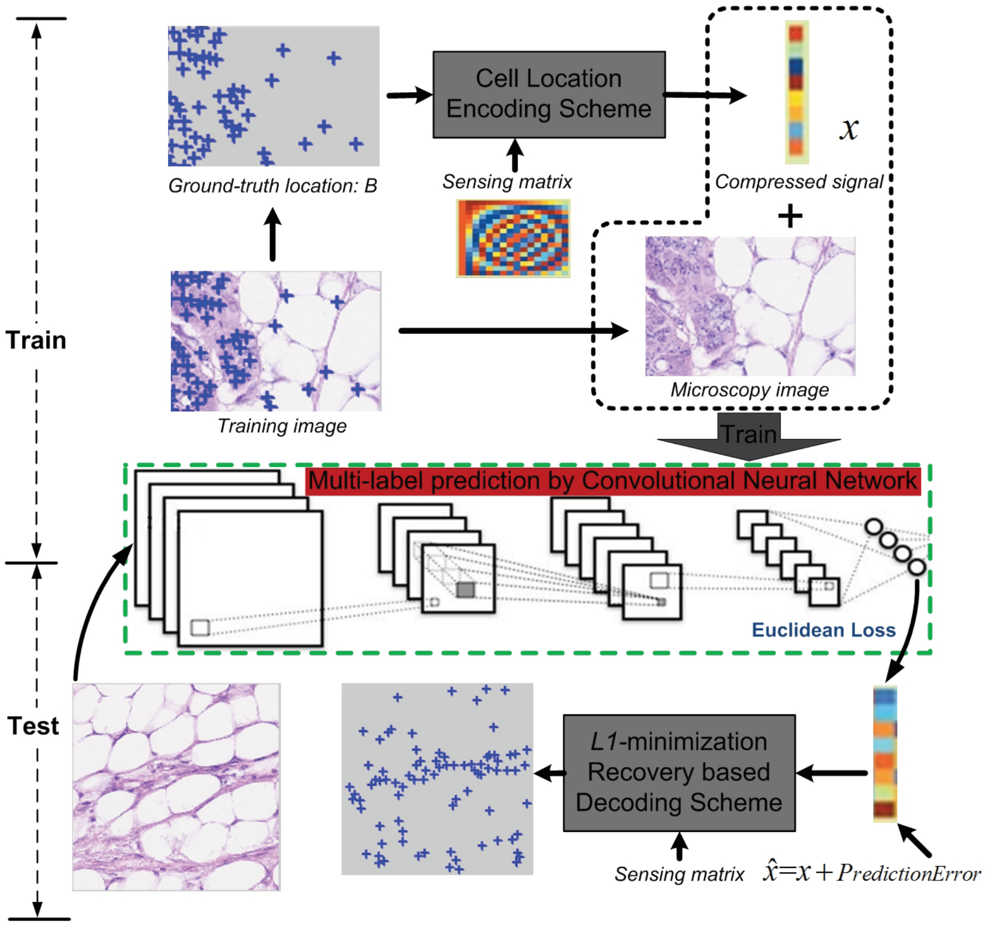

Train
Test
Ground-truth location: B
Cell Location Encoding Scheme
Sensing matrix
x
Compressed signal
+
Training image
Microscopy image
Train
Multi-label prediction by Convolutional Neural Network
Euclidean Loss
L1-minimization Recovery based Decoding Scheme
Sensing matrix
$\hat{x}=x+PredictionError$

Figure 4.3: The system overview of the proposed CNNCS framework for cell detection and localization.

provided the sensing matrix D satisfies a restricted isometry property (RIP) condition [67], [26]. In many cases, D is typically a $m \times n$ ($m \ll n = hw$) random Gaussian matrix.

Scheme-2: Encoding by Projection

For the encoding scheme-1, the space complexity of the interim result a is $\mathcal{O}(wh)$. For example, to encode the location of cells in a 260-by-260 pixel image, scheme-1 will produce a as a 67,600-length vector; so that in the subsequent CS process, a huge sensing matrix in size of m-by-67600 is required in order to match the dimension of a, which will make the system quite slow, even unacceptable for larger images. To further optimize the encoding scheme, we propose a second scheme, where the coordinates of every cell centroid are projected onto multiple observation axes. We refer the second encoding scheme as "Scheme-2: Encoding by Projection."

To encode location of cells, we create a set of observation axes $OA = \{oa_l\}, l = 1, 2, \ldots, L$, where L indicates the total number of observation axes used. The observation axes are uniformly-distributed around an image (See Figure 4.4, left-most picture) For the l-th observation axis oa_l, the location of cells is encoded into a R-length ($R = \sqrt{w^2 + h^2}$) sparse signal, referred as a_l (See Figure 4.4, third picture). We calculate the perpendicular signed distances (a_l) from cells to oa_l. Thus, a_l contains signed distances, which not only measure the distance, but also describe on which side of oa_l cells are located. After that, the encoding of cell locations under oa_l is x_l, which is obtained by the following random projection:

$$x_l = Da_l, \tag{4.5}$$

where, D is typically a $m \times n$ random Gaussian matrix. Here, the number of observations m is much smaller than n. We repeat the above process for all the L observation axes and obtain each x_l. After concatenating all the x_l, $l = 1, 2, \ldots, L$, the final encoding result x is available, which is the joint representation of cells location. The whole encoding process is illustrated by Figure 4.4.

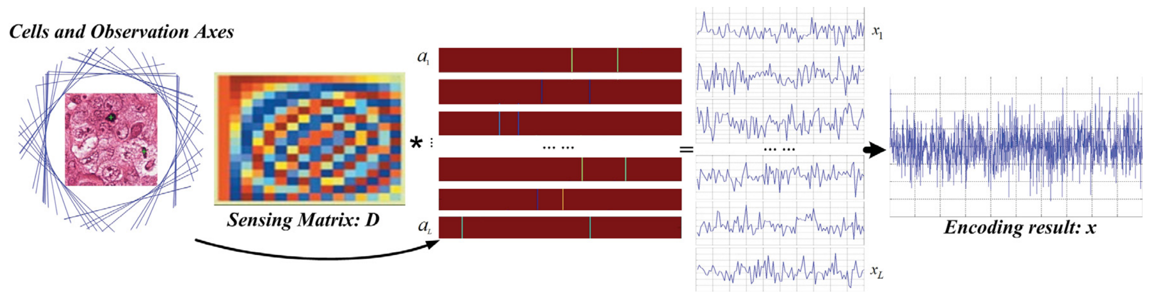


Figure 4.4: Cell location encoding by signed distances (Scheme-2).

For encoding scheme-2, the size of the sensing matrix D is m-by-$\sqrt{w^2 + h^2}$. In comparison, encoding scheme-1 requires a much larger sensing matrix of size m-by-wh. The first advantage of encoding scheme-2 is that it dramatically reduces the size of the sensing matrix, which is quite helpful for the recovery process, especially when the size of images is large. Secondly, the encoding result x carries **redundant** information about cell locations. In the subsequent decoding phase, averaging over the redundant information makes the final detection more reliable. More details can be found in experiments section. A final point is that in case more than one cell locations are projected to the same bin in a particular observation axis, such a conflict will not occur for the same set of cells at other observation axes.

4.6.2 Decoding Scheme

Recovery of a_l can be estimated from the predicted compressed signal $\hat{x_l}$ by solving the following L_1 norm convex optimization problem:

$$\hat{a_l} = \underset{a}{\operatorname{argmin}} \|a\|_1 \qquad \text{subject to} \qquad \hat{x_l} = Da_l \tag{4.6}$$

After $\hat{a_l}$ is recovered, every true cell is localized L times, i.e. with L candidate positions predicted. The redundancy information allows us to estimate more accurate detection of a true cell.

The first two images of Figure 4.5 from left present examples of the true location signal a and decoded location signal $\hat{a}$, respectively. The noisy signed distances of $\hat{a}$ are typically very close to each observation axis. That is why we create observation axes outside of the image space, so that these noisy distances can be easily distinguished from true candidate distances. This separation is done by mean shift clustering, which also groups true detections into localized groups of detections. Two such groups (clusters) are shown in Figure 4.5, where the signed distances formed circular patterns of points (in green) around ground truth detections (in yellow). Averaging over these green points belonging to a cluster provides us a predicted location (in red) as shown in Figure 4.5.

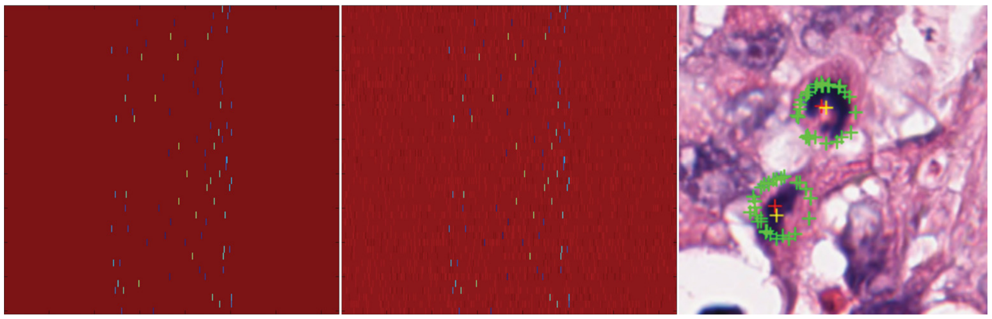


Figure 4.5: Cell Location Decoding Scheme. From left to right: true location signal a, decoded location signal $\hat{a}$ and detection results. Yellow crosses indicate the ground-truth location of cells, green crosses are the candidates points, red crosses represent the final detected points.

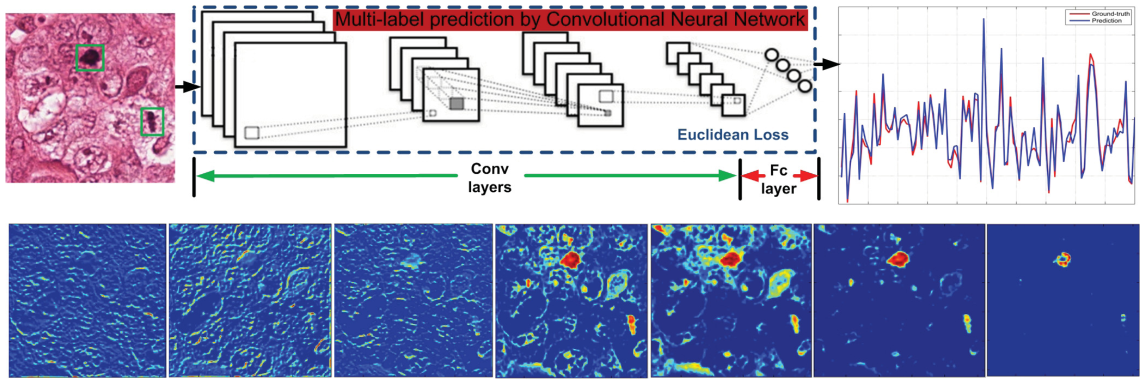


Figure 4.6: An illustration of the process of signal prediction by convolutional neural network. The bottom row presents the feature maps learned from Convolutional (Conv) layers of the CNN with training process going on. The current CNN follows the AlexNet architecture. These feature maps come from the Conv1, Conv1, Conv2, Conv3, Conv3, Conv4 and Conv5 respectively. The top-right picture shows the ground-truth compressed signal (red) and compressed signal (blue) predicted from the Fullly-connected (Fc) layer of the CNN. From the picture, we can observe that the predicted signal approximate the pattern of ground truth signal well.

4.7 Signal Prediction by Convolutional Neural Network

We utilize a CNN to build a regression model between a cell microscopy image and its cell location representation: compressed signal y. We employ two kinds of CNN architectures. One of them is AlexNet [52], which consists of 5 convolution layers + 3 fully connected layers; the other is the deep residual network (ResNet) [36] where we use its 152-layer model. In both the architectures, the loss function is defined as the Euclidean loss. The dimension of output layer of AlexNet and ResNet has been modified to the length of compressed signal y. We train the AlexNet model from scratch, in comparison, we perform fine-tuning on the weights in fully-connected layer of the ResNet.

To prepare the training data, we generate a large number of square patches from training images. Along with each training patch, there is a signal (i.e. the encoding result: y), which indicates the location of target cells present in the patch. After that, patch rotation is performed on the collected training patches for data augmentation and making the system rotation invariant.

The trained CNN not only predicts the signal from its output layer, the feature maps learned from its Conv layers also provide rich information for recognition. Figure 4.6 visualizes the learned feature maps, which represents the probabilistic score or activation maps of target cell regions (indicated by green boxes in the left image) during training process. It can be observed that higher scores are fired on the target regions of score masks, while most of the non-target regions have been suppressed more and more with training process going on.

To further optimize our CNN model, we apply Multi-Task Learning (MTL) [11]. During training a CNN, two kinds of labels are provided. The first kind is the encoded vector: x, which carries the pixel-level location information of cells. The other kind is a scalar: cell count (c), which indicates the total number of cells in a training image patch. We concatenate the two kinds of labels into the final training label by $label = \{x, \lambda c\}$, where λ is a hyper parameter. Then, Euclidean loss is applied on the fusion label. Thus, supervision information for both cell detection and cell counting can be jointly used to optimize the parameters of our CNN model.

4.8 Theoretical Justification

Equivalent Targets for Optimization

We first show that from the optimization standpoint, compressed vector is a good proxy for the original, sparse output space. This result directly follows from the CS theory. As mentioned before, a indicates the cell location represented in pixel space, and x is the cell location represented in compressed signal space. They follow the relationship: $x = Da$, where D is the sensing matrix. Let us assume that a_p and a_g are respectively the prediction and ground-truth vectors in the pixel space. Similarly, we have x_p and x_g as their compressed counterparts, respectively.

Claim: $\|x_g - x_p\|$ and $\|a_g - a_p\|$ are approximately equivalent targets for optimization.

Proof: According to the CS theory, a sensing matrix $D \in \mathbb{R}^{m \times d}$ should satisfy the $(k, \delta) - restricted\ isometry\ property\,((k, \delta) - RIP)$, which states that for all $k-$sparse $a \in \mathbb{R}^d$, $\delta \in (0, 1)$, the following holds [67], [26], [23]:

$$(1 - \delta)\,\|a\| \leq \|Da\| \leq (1 + \delta)\,\|a\|. \tag{4.7}$$

Note that if the sensing matrix D satisfies $(2k, \delta)$-RIP, then (4.7) also holds good. Now replace a with $(a_g - a_p)$ and note that $(a_g - a_p)$ is $2k$-sparse. Thus,

$$(1 - \delta)\,\|a_g - a_p\| \leq \|x_g - x_p\| \leq (1 + \delta)\,\|a_g - a_p\|\,. \tag{4.8}$$

From the right hand side inequality, we note that if $\|a_g - a_p\|$ is small, then $\|x_g - x_p\|$ would be small too. In the same way, if $\|x_g - x_p\|$ is large, then the inequality implies that $\|a_g - a_p\|$ would be large too. Similarly, from the left hand side inequality, we note that if $\|a_g - a_p\|$ is large then $\|x_g - x_p\|$ will be large, and if $\|x_g - x_p\|$ is small then $\|a_g - a_p\|$ will small too. These relationships prove the claim that from the optimization perspective $\|x_g - x_p\|$ and $\|a_g - a_p\|$ are approximately equivalent.

A Bound on Generalization Prediction Error

In this section we mention a powerful result from [37]. Let x be the predicted compressed vector by the CNN, a be the ground truth sparse vector, $\hat{a}$ be the reconstructed sparse vector from prediction, and D be the sensing matrix. Then the generalization error bound provided in [37] is as follows:

$$\|\hat{a} - a\|_2^2 \leq C_1 \cdot \|x - Da\|_2^2 + C_2 \cdot sperr(\hat{a}, a), \tag{4.9}$$

where C_1 and C_2 are two small constants and *sperr* measures how well the reconstruction algorithm has worked [37]. This result demonstrates that expected error in the original space is bound by the expected errors of the predictor and that of the reconstruction algorithm. Thus, it makes sense to apply a very good machine learner such as deep CNN that can minimize the first term in the right hand side of (4.9). On the other hand, DAL provides one of the best L_1 recovery algorithms to minimize the second term in the right side of (4.9).

4.9 Experiments

4.9.1 Datasets and Evaluation Criteria

We utilize seven cell datasets, on which CNNCS and other comparison methods are evaluated. The 1st dataset [46] involves 100 H&E stained histology images of colorectal adenocarcinomas. The 2nd dataset [10] consists of 200 highly realistic synthetic emulations of fluorescence microscopic images of bacterial cells. The 3rd dataset [102] comprises of 55 high resolution microscopic images of breast cancers double stained in red (cytokeratin epithelial marker) and brown (nuclear proliferative marker). The 4th dataset is the ICPR 2012 mitosis detection contest dataset [77] including 50 high-resolution (2084-by-2084) RGB microscope slides of Mitosis. The 5th dataset [2] is the ICPR 2014 grand contest of mitosis detection, which is a follow-up and an extension of the ICPR 2012 contest on detection of mitosis. Compared with the contest in 2012, the ICPR 2014 contest is much more challenging, which contains a much larger number of images for training and testing. The 6th dataset is the AMIDA-2013 mitosis detection dataset [94], which contains 676 breast cancer histology images belonging to 23 patients. The 7th dataset is the AMIDA-2016 mitosis detection dataset [1], which is an extension of the AMIDA 2013 contest on detection of mitosis. It contains 587 breast cancer histology images belonging to 73 patients for training, and 34 breast cancer histology images for testing with no ground truth available. For each dataset, the annotation that represents the location of cell centroids is shown in Figure 4.7, details of datasets are summarized in Table 4.1.

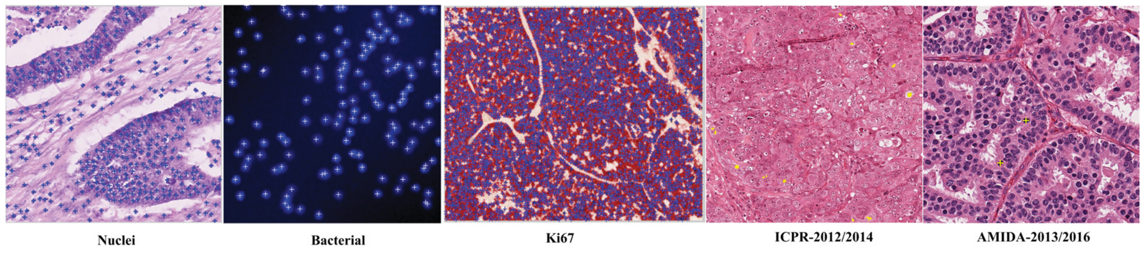


Figure 4.7: Dataset examples and their annotation.

Table 4.1: *Size* is the image size; *Ntr/Nte* is the number of images selected for training and testing; AC indicates the average number of cells per image.

Cell Dataset	Size	Ntr/Nte	AC
Nuclei [46]	500×500	50/50	310.22
Bacterial [10]	256×256	100/100	171.47
Ki67 Cell [102]	1920×2560	45/10	2045.85
ICPR 2012 [77]	2084×2084	35/15	5.31
ICPR 2014 [2]	1539×1376	1136/496	4.41
AMIDA 2013 [94]	2000×2000	447/229	3.54
AMIDA 2016 [1]	2000×2000	587/34	2.13

For evaluation, we adopt the criteria of the ICPR 2012 mitosis detection contest [77], which is also adopted in several other cell detection contests. A detection would be counted as true positive (TP) if the distance between the predicted centroid and ground truth cell centroid is less than ρ. Otherwise, a detection is considered as false positives (FP). The missed ground truth cells are counted as false negatives (FN). In our experiments, ρ is set to be the radius of the smallest cell in the dataset. Thus, only centroids that are detected to lie inside cells are considered correct. The results are reported in terms of Precision: $P = TP/(TP + FP)$ and Recall: $R = TP/(TP + FN)$ and F_1-score: $F_1 = 2PR/(P + R)$ in the following sections.

4.9.2 Experiments with Encoding Scheme-1

To evaluate, we carry out performance comparison experiment between CNNCS and three state of the art cell detection methods ("FCN-based" [101], "Le.detect" [6], "CasNN" [13]). In this experiment, the scheme-1: encoding by reshaping is applied in CNNCS.

For the four methods to provide different values of Precision-Recall as shown in Figure 4.8, we tune hyper parameters of every method. With scheme-1, CNNCS has a threshold T to apply on the recovered sparse signal $\hat{a}$ before re-shaping it to a binary image B. T is used to perform cell vs. non-cell binary classification and can be treated as a hyper parameter during training. In "FCN-based" [101], there is also a threshold applied to the local probability-maximum candidate points to make final decision about cell or non-cell. Similarly, in the first step of "Le.detect" [6], researchers use a MSER-detector (a stability threshold involved here) to produce a number of candidate regions, on which their learning procedure determines which of these candidates regions correspond to cells. In the first experiment, we analyze the three methods using Precision-Recall curves by varying their own thresholds.

Figure 4.8 presents Precision-Recall curves on three cell datasets. All the four methods give reliable detection performances in the range of recall=[0.1-0.4]. After about recall=0.6, the precision of "FCN-based" [101] drops much faster. This can be attributed to the fact that "FCN-based" [101] works by finding local maximum points on a cell density map. However, the local maximum operation fails in several scenarios, for example when two cell density peaks are close to each other, or large peak may covers neighboring small peaks. Consequently, to obtain the same level of recall, "FCN-based" [101] provides many false detections.

Furthermore, it also can be observed that CNNCS has an improvement over "Le.detect" [6] (red line clearly outperforms black line under varying recall values). This can be largely explained by the fact that traditional methods (no matter if [6] or [101] is used) always try to predict the coordinates of cells directly on a 2-D image. The coordinates are sensitive to system prediction bias or error, considering the nature of cell detection that cells are small and quite dense in most cases. It is not surprising that "Le.detect" [6] will miss some cells and/or detect other cells in wrong locations. In comparison, CNNCS transfers the cell detection task from pixel space to compressed signal space, where the location information of cells is no longer represented by $\{i, j\}$-coordinates. Instead, CNNCS performs cell detection by regression and recovery on a fixed length compressed signal. Compared to $\{i, j\}$-coordinates representation, the compressed signal is more robust to system prediction errors. For example, as shown in the right top corner of Figure 4.6, even though there are differences between the ground-truth compressed signal and predicted compressed signal, the whole system can still give reliable detection performance as shown in Figure 4.8.

To get a better idea of the CNNCS method, we visualize a set of cell images with their detected cells and ground-truth cells in Figure 4.9. It can be observed that CNNCS is able to accurately detect most cells under a variety of conditions.

4.9.3 Experiments with Encoding Scheme-2

The influence of several hyperparameters to overal performance

Table 4.2: The influence of M to overal performance.

M	Precision	Recall	F_1-score
45	0.3937	0.5972	0.4744
59	0.4896	0.6128	0.5443
89	0.6671	0.6954	0.6810
74	0.5864	0.7202	0.6465
96	0.6358	0.8604	0.7312
102	0.6712	0.8778	0.7607
112	0.8720	0.8052	0.8371
125	0.6513	0.8363	0.7323
132	0.4756	0.8487	0.6096
152	0.4775	0.7581	0.5859
165	0.4896	0.6753	0.5676

Based on the above experiments, we acquire some basic understanding of the influence of each important hyper parameter on the overall detection accuracy. Then we perform random search by choosing a narrow ranges for the hyper parameters, and tune them jointly. After that, we obtain a group of hyper parameters and cooresponding cell detection result as shown in Table 4.5.

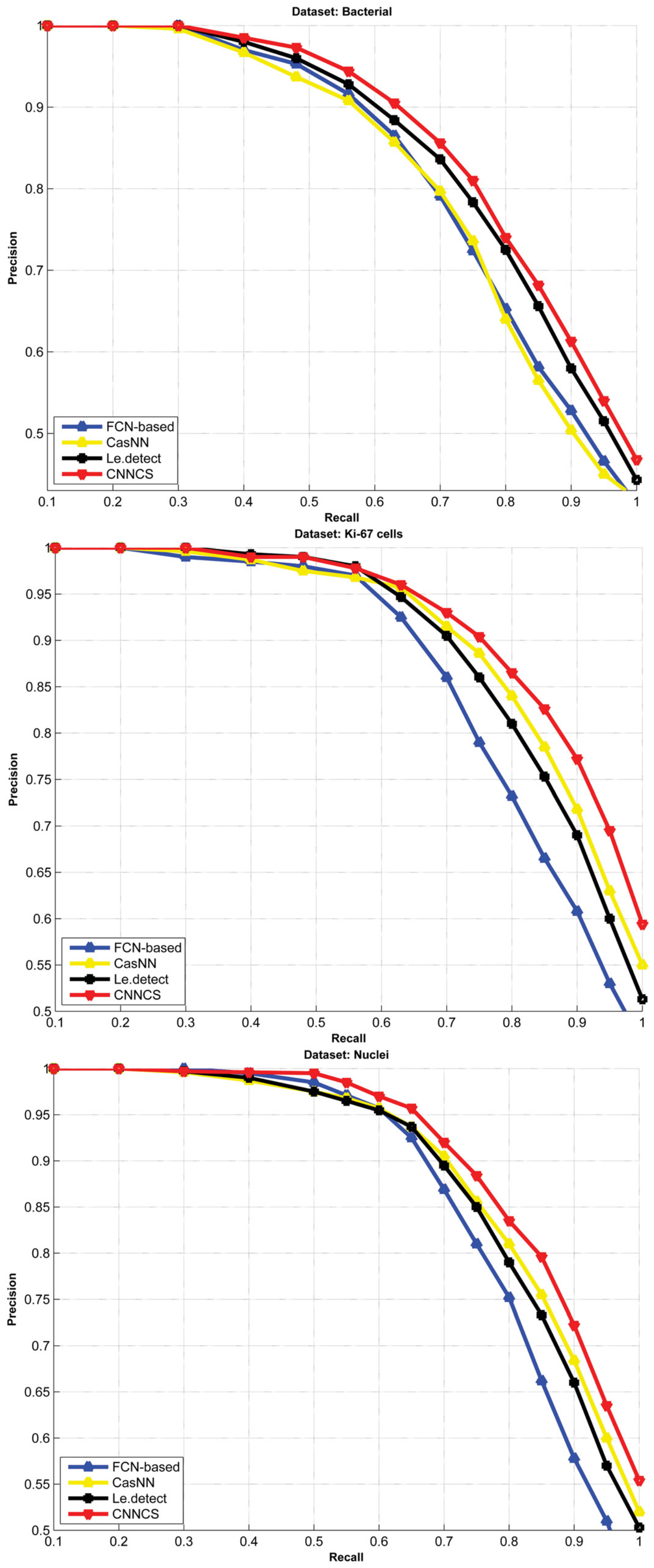


Figure 4.8: Precision and recall curves of four methods on three datasets.

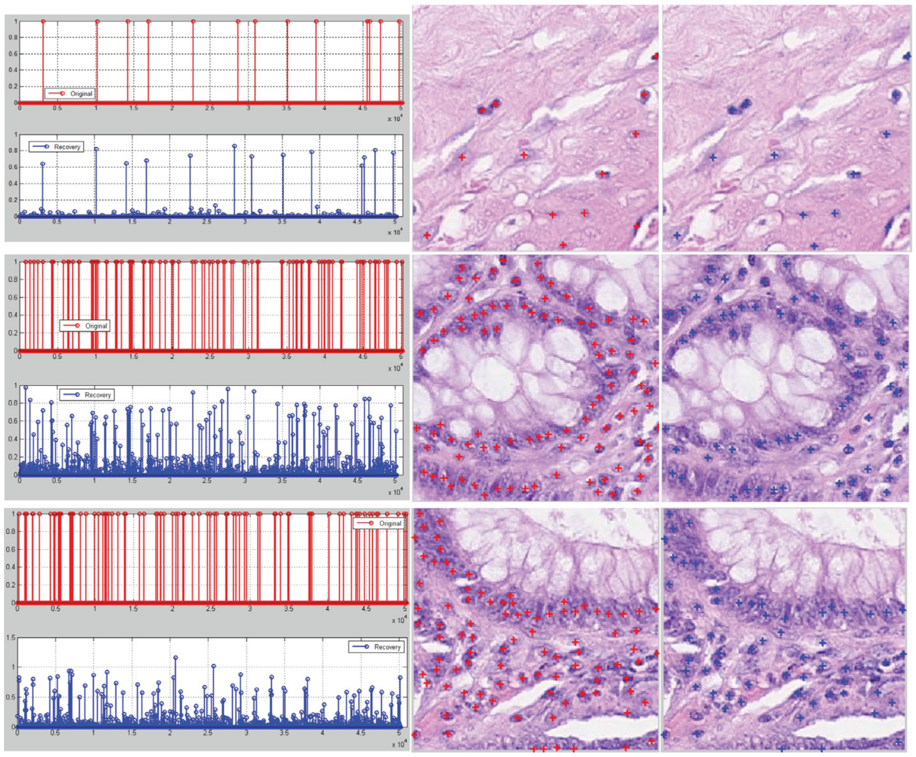


Figure 4.9: Detection results. Ground-truth: red, Prediction: blue. Left part shows the ground truth signal and the predicted sparse signal that carries the location information of cells; right part shows the ground-truth and detected cells.

Experiment on ICPR 2012 mitosis detection dataset

To evaluate the performance of encoding scheme 2, we carry out the second group of performance comparison experiments. In the first experiment, we apply the proposed method on the ICPR 2012 mitosis detection contest dataset, which consists of 35 training images and 15 testing images. For the training process, we extracted image sub-samples (260-by-260) with no overlap between each other from the 35 training images. After that every 90° image rotation is performed on each sub-sample for data augmentation, resulting in a total of 8,960 training dataset. In addition, we perform random search to tune the three hyper parameters in scheme-2: (1) the number of rows in sensing matrix: m, (2) the number of observation lines: L and (3) the importance (λ) of cell count during MTL. After that, the best performance is achieved when $m = 112, L = 27, \lambda = 0.20$. Furthermore, we trained five CNN models to reduce the performance variance introduced by a single model and to improve the robustness of the whole system. Recently, deep residual network (ResNet) introduces residual connections into deep convolutional networks and has yielded state of the art performance in the 2015 ILSVRC challenge [36]. This raises the question of whether there is any benefit in introducing and exploiting more recent CNN architectures into the cell detection task. Thus, in the experiment, we have explored the performance of CNNCS with different neural network architectures (AlexNet and ResNet). Finally, CNNCS gets the

Table 4.3: The influence of L to overal performance.

L	Precision	Recall	F_1-score
6	0.4390	0.5279	0.4793
15	0.5194	0.6753	0.5872
27	0.7040	0.8896	0.7860
38	0.7229	0.8214	0.7690
46	0.5595	0.8584	0.6774
58	0.5362	0.6825	0.6024
72	0.5276	0.6851	0.5961

Table 4.4: The influence of λ to overal performance.

λ	Precision	Recall	F_1-score
0.10	0.4204	0.7151	0.5295
0.15	0.5158	0.8989	0.6555
0.20	0.6224	0.8750	0.7275
0.25	0.5310	0.7477	0.6210
0.30	0.4430	0.7267	0.5505

highest F1-score among all the comparison methods, details are summarized in Table 4.6.

Compared to the state of the art method: CasNN-average [13], CNNCS with ResNet and MTL achieved a better performance with F_1-score 0.837. It can be observed from Table 4.6 that the precision of our method outperforms the previous best precision by 0.06-0.07, and recall also has recorded about 0.02 improvement. This phenomenon can be attributed to the detection principle of our method, where every ground-truth cell is localized with multiple candidate points guaranteed to be around the true location, then the average coordinates of these candidates is computed as the final detection. As a result, localization closer to the true cell becomes more reliable compared to other methods, thus leading to a higher precision. In addition, an improvement of F_1-score from 0.833 to 0.837 achieved by MTL demonstrates that the knowledge jointly learned from cell detection and cell counting provides further benefits at negligible additional computations.

Experiment on ICPR 2014 mitosis detection dataset

In the second experiment, we evaluated CNNCS on the ICPR 2014 contest of mitosis detection dataset (also called MITOS-ATYPIA-14), which is a follow-up and an extension of the ICPR 2012 contest on detection of mitosis. Compared with the contest in 2012, the ICPR 2014 contest is much more challenging, which contains more images for training and testing. It provides 1632 breast cancer histology images, 1136 images for training, 496 images for testing. Each image is in the size of 1539×1376. We divide the training images into training set (910 images) and validation set (226 images). We perform random search on the validation set to optimize the hyper parameters. The best performance on MITOS-ATYPIA-14 dataset is achieved when $m = 103, L = 30, \lambda = 0.24$. On the test dataset, we have achieved the **highest** F1-score among all the participated teams. The F1-score of

Table 4.5: Hyperparameters tuning by random search on validation set of AMIDA 2016 dataset.

M	L	λ	Precision	Recall	F_1-score
118	31	0.20	0.5691	0.6781	0.6188
115	35	0.29	0.6149	0.6531	0.6340
107	24	0.21	0.5312	0.7408	0.6188
105	20	0.24	0.5568	0.7321	0.6326
106	39	0.18	0.5570	0.6679	0.6075
121	29	0.25	0.6373	0.6104	0.6236

Table 4.6: Results on testing set of ICPR 2012 grand challenge of mitosis detection.

Method	Precision	Recall	F_1-score
UTRECHT	0.511	0.680	0.584
NEC [63]	0.747	0.590	0.659
IPAL [40]	0.698	0.740	0.718
DNN [15]	0.886	0.700	0.782
RCasNN [13]	0.720	0.713	0.716
CasNN-single [13]	0.738	0.753	0.745
CasNN-average [13]	0.804	0.772	0.788
CNNCS-AlexNet	0.860	0.788	0.823
CNNCS-ResNet	0.867	0.801	0.833
CNNCS-ResNet-MTL	0.872	0.805	**0.837**

all the participated teams are shown in Table 4.7. As we see, the CNNCS method shows significant improvement compared to the results of other teams in all the histology slice groups. On an average, CNNCS has almost doubled the F1-score of teams at the second place.

Table 4.7: Results on testing set of ICPR 2014 contest of mitosis detection.

Slice group	CUHK	MINES	YILDIZ	STRAS	CNNCS
A06	0.119	0.317	0.370	0.160	**0.783**
A08	0.333	0.171	0.172	0.024	**0.463**
A09	0.593	0.473	0.280	0.072	**0.660**
A19	0.368	0.137	0.107	0.011	**0.615**
Average	0.356	0.235	0.167	0.024	**0.633**

Experiment on AMIDA 2013 mitosis detection dataset

The third experiment was performed on the AMIDA-2013 mitosis detection dataset, which contains 676 breast cancer histology images, belonging to 23 patients. Suspicious breast tissue is annotated by at least two expert pathologists, to label the center of each cancer cell. We train the proposed CNNCS method using 377 images, validate on 70 training images and test it on the testing set of AMIDA-2013 challenge that has 229 images from the last 8 patients. We employ ResNet as the network architecture with data balancing and MTL in the training set. Similar to previous experiments, we perform random search on the validation set to optimize the hyper parameters. The best performance on AMIDA-2013 dataset is achieved when $m = 118, L = 25, \lambda = 0.32$. Finally among all the 17 participated teams, we achieve the **third highest** F1-score=0.471, which is quite close to the second place, and has a significant improvement over the fourth place method [80]. For details, Table 4.8 summarizes the comparison between CNNCS and other methods.

Table 4.8: Results on testing set of AMIDA-2013 MICCAI grand challenge of mitosis detection.

Method	Precision	Recall	F_1-score
IDSIA [15]	0.610	0.612	0.611
DTU	0.427	0.555	0.483
AggNet [80]	0.441	0.424	0.433
CUHK	0.690	0.310	0.427
SURREY	0.357	0.332	0.344
ISIK	0.306	0.351	0.327
PANASONIC	0.336	0.310	0.322
CCIPD/MINDLAB	0.353	0.291	0.319
WARWICK	0.171	0.552	0.261
POLYTECH/UCLAN	0.186	0.263	0.218
MINES	0.139	0.490	0.217
SHEFFIELD/SURREY	0.119	0.107	0.113
SEOUL	0.032	0.630	0.061
UNI-JENA	0.007	0.077	0.013
NIH	0.002	0.049	0.003
CNNCS	0.3588	0.5529	0.4352

Experiment on AMIDA 2016 mitosis detection dataset

In the fourth experiment, we participated in the AMIDA-2016 mitosis detection challenge (also called TUPAC16), which is a follow-up and an extension of the AMIDA-2013 contest on detection of mitosis. Its training dataset has 587 breast cancer histology images in size of 2000×2000, belonging to 73 patients. Its test dataset contains 34 breast cancer histology images in the same size without publicly available ground truth labels.

We train the proposed CNNCS method using randomly chosen 470 training images and validate on the remaining 117 training images. Additionally, we apply the following ensemble averaging technique to further increase precision and recall values. Originally, we have partitioned every test image into about 100 non-overlapping patches. Instead of starting the partitioning from the top-left corner, now we set the starting point of the first patch from {offset, offset}. The offset values are set as 0, 20, 40,..., 160, and 180 (i.e. every 20 pixel) resulting in a total of 10 different settings. Under every offset setting, CNNCS method is run on all the generated patches and provides detection results. Then, we merge detection results from all the offset settings. The merging decision rule is that if there are 6 or more detections within a circle in radius of 9 pixels, then we accept average of these locations as our final detected cell center. Other implementation settings are similar to the settings in the experiment of AMIDA-2013. The best performance on AMIDA-2016

Table 4.9: Results of AMIDA-2016 MICCAI grand challenge of mitosis detection (my method's result is on the validation set).

Team	F_1-score
Lunit Inc.	0.652
IBM Research Zurich and Brazil	0.648
Contextvision (SLDESUTO-BOX)	0.616
The Chinese University of Hong Kong	0.601
Microsoft Research Asia	0.596
Radboud UMC	0.541
University of Heidelberg	0.481
University of South Florida	0.440
Pakistan Institute of Engineering and Applied Sciences	0.424
University of Warwick	0.396
Shiraz University of Technology	0.330
Inha University	0.251
CNNCS (on validation set)	0.634

dataset is achieved when $m = 115, L = 35, \lambda = 0.29$. Finally, we achieved F1-score=0.634 on the validation set (becuase of the lack of publicly available test set), which is the **third highest** in all the 15 participated teams. Table 4.9 provides more details of the contest results. Furthermore, Figure 4.10 provides twelve examples of our detection results in the AMIDA-2016 grand challenge of mitosis detection.

4.9.4 Experimental justification for compressed sensing

Training a regression model without random projection

To better understand the meaning of applying output space encoding, we perform an experiment, where we do not use random projections. Thus, the representation of true cell locations stays in the pixel space. We still use cell location encoding scheme 2, which encodes the i-j coordinates of cells to a sparse code a, which indicates the signed distances from cells to observation lines. We train the same CNN based regression model with the same architecture as before. During testing, the trained CNN based regression model will estimate a sparse code $\hat{a}$, from which predicted cells' i-j coordinates are predicted. We refer this CNN based regression model as "CNN-only".

Figure 4.11 visualizes three typical examples of cell detection results given by the CNN-only. The prediction errors from the trained CNN-only model results in the position shift of detected points (green points in Figure 4.11). Previously, in the CNNCS model, with the constrain from L_1 minimization based decoding, a predicted point (green point) is typically closer to an observation line than ground-truth point, because its total distance (can be regarded as kind of energy) has been consumed by small noisy points. To look at a higher level, these results in the phenomenon that predicted points by observation lines form a cluster, which is almost around a ground-truth point (red point), as shown in Figure 4.10. Then we perform meanshift clustering to find the final detection (red

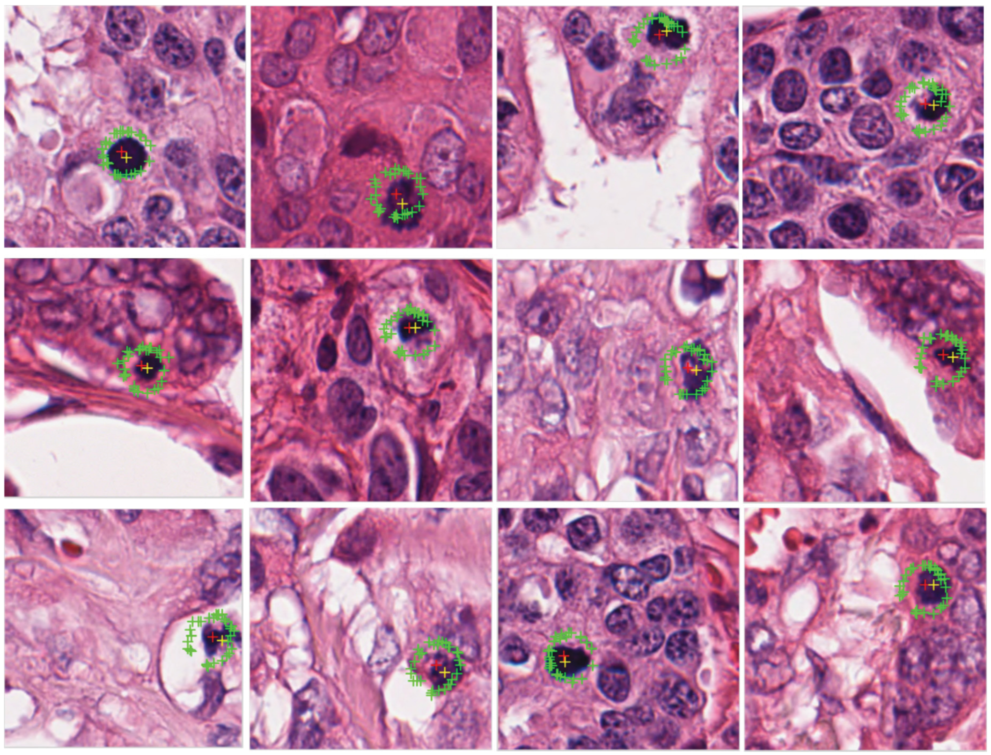


Figure 4.10: Results on AMIDA-2016 dataset. Yellow cross indicates the ground-truth position of target cells. Green cross indicates cell position predicted by each observation axis. Red cross indicates the final detected cell position, which is the clustering center of all green crosses.

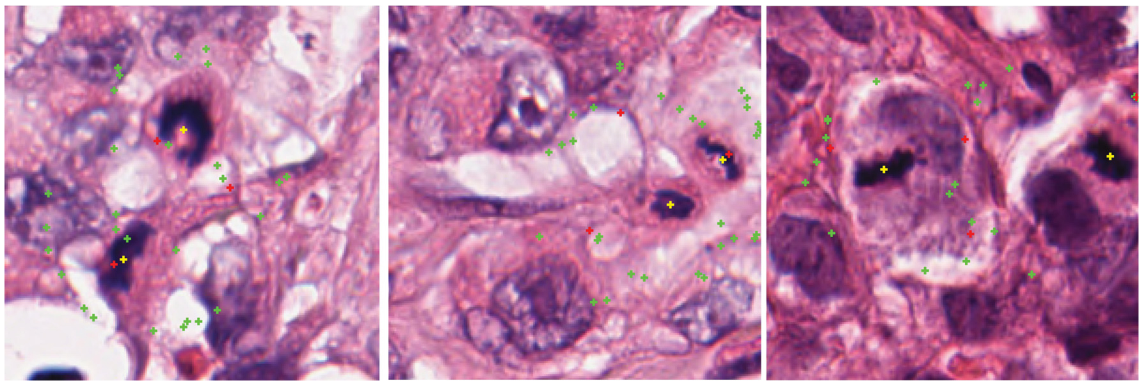


Figure 4.11: Cell detection results by the CNN-only model. Yellow cross indicates the ground-truth position of target cells. Green cross indicates cell position predicted by each observation axis. Red cross indicates the final detected cell position, which is the clustering center of all green crosses.

point). But for CNN-only, there is no such structured property among these predicted points (green points). They are the response of every strong element of a sparse code in the pixel space. And there is no correlations or constrain between these strong elements (unlike the constrain from L_1 minimization based decoding). Furthermore, the sparse code is predicted by the CNN based regression model, and is directly affected by the CNN prediction errors. An error in the CNN output vector will directly result in pixel shifts of predicted points. All these observations serve as strong motivations to apply CS-based output space encoding in the CNNCS model. We compare accuracies of CNNCS and CNN-only models on four benchmark datasets mentioned before. Precision, Recall and F_1-scores are summarized in Table 4.10-Table 4.13. The quantitative performance comparison further verify the importance of introducing CS-based output space encoding.

Table 4.10: Results of CNNCS and CNN-only on validation set of AMIDA-2016 dataset.

Method	Precision	Recall	F_1-score
CNN-only	0.3271	0.4089	0.3635
CNNCS	0.5646	0.7240	0.6344

Table 4.11: Results of CNNCS and CNN-only on testing set of AMIDA-2013 dataset.

Method	Precision	Recall	F_1-score
CNN-only	0.2361	0.3180	0.2710
CNNCS	0.3588	0.5529	0.4352

Table 4.12: Results of CNNCS and CNN-only on testing set of ICPR-2014 dataset.

Method	Precision	Recall	F_1-score
CNN-only	0.3408	0.4351	0.3822
CNNCS	0.6157	0.6510	0.6329

4.10 Summary of this chapter

This is the first attempt to demonstrate that deep convolutional neural network can work in conjunction with compressed sensing-based output encoding schemes toward solving a significant medical image processing task: cell detection and localization from microscopy images. In this work, we made substantial experiments on several benchmark datasets and challenging cell detection contests, where the proposed CNN + CS framework (referred to as CNNCS) achieved very competitive (the highest or at least top-3 in terms of F1-score) results compared to the state of the art methods in cell detection task. In addition, the CNNCS framework has the potential to be End-to-End trained, which is described in the next chapter.

Table 4.13: Results of CNNCS and CNN-only on testing set of ICPR-2012 dataset.

Method	Precision	Recall	F_1-score
CNN-only	0.411	0.443	0.426
CNNCS	0.872	0.805	0.837

Chapter 5

End-to-End Training of CNN with Compressed Sensing

5.1 Introduction

Chapter 4 provides a detailed description on our proposed CNNCS framework for cell detection. The CNNCS framework consists of two components, one is a CNN based regressor, the other is a CS-based decoder that predicts a sparse vector.

During training, only the CNN based regressor is trainable. During testing, the two components are used in a cascaded processing pipeline. As mentioned before, the CNN based regressor is responsible for estimating a compact and dense code $\hat{x}$. Any deviations between the the predicted code $\hat{x}$ and ground truth x are the prediction error of the first module. This error will pass to the second module and affect the accuracy of the recovered sparse code predicted by the second module. The principal limitation of CNNCS model is the lack of adaptation of the CS-based decoder to the noisy signal predicted by the CNN.

To tackle this problem, in this chapter, we propose a refined CNNCS framework which renders the cascaded structure of CNNCS end-to-end trainable. In the end-to-end trainable model, the loss value from the very end will be back propagated through the whole architecture and the parameters of both the CNN based regressor and the CS based sparse code predictor can be adjusted jointly during back propagation. Thus, the CS decoder can adapt to the noisy prediction of CNN and vice-versa. In this chapter, we first introduce a trainable sparse code prediction algorithm, called LISTA [33], as a CS decoder. Next, we derive a backpropagation rule for a CS decoder that is to the best of our knowledge the first such use of backpropagation across a sparse coding / decoding layer in an end-to-end architecture.

5.2 System Overview

Different from the CNNCS model that consists of two cascaded components, a CNN based regressor and a CS based sparse code predictor, the end-to-end trainable model is a single neural network. This end-to-end model is for the perpose of joint optimization of all the parameters. The first *ol* layers of the end-to-end trainable model are observation layers, which encode an input image into a dense code x via a convolutional neural network structure. The dense code x forms the representation of the locations of target cells in an image. *ol* can make use of a particular CNN architecture, such as, AlexNet or ResNet. The subsequent *rl* layers are reconstruction layers, which follow the architecture of the trainable sparse code predictor described in the next section. We implement the trainable sparse code predictor as the reconstruction layers and insert it into the whole architecture. The reconstruction layers perform L_1 minimization to reconstruct a sparse code a for each dense code x. Then the sparse code a can be simply decoded back to the location of target cells in the pixel space.

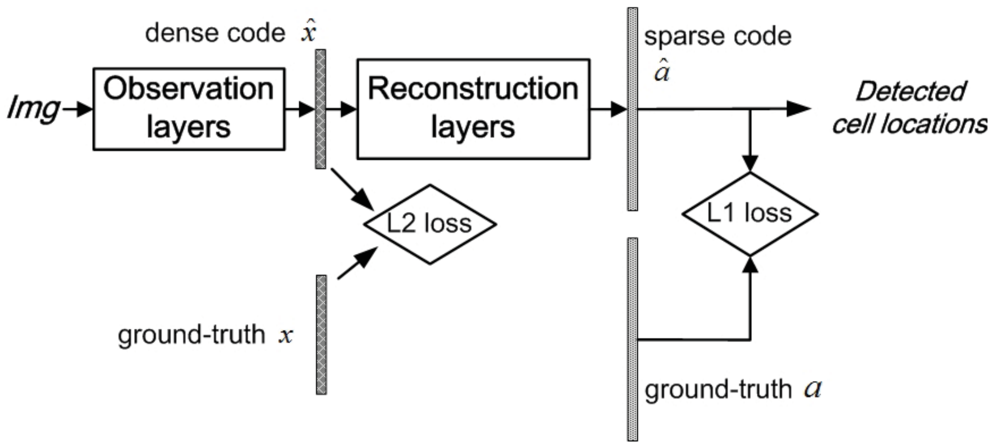


Figure 5.1: System overview of the proposed end-to-end trainable model for cell detection.

Figure 5.1 illustrates the system overview of the end-to-end trainable model, where the dense code $\hat{x}$ and the sparse code $\hat{a}$ are the outputs of *ol* and *rl*, respectively. x and a are the ground-truth codes corresponding to $\hat{x}$ and $\hat{a}$. We apply L_1 norm loss to measure the error between sparse codes a and $\hat{a}$, and apply L_2 norm loss to measure the error between dense codes x and $\hat{x}$.

We develop two versions of the end-to-end trainable model. In the first version, we use LISTA [33] as *rl*. Because, LISTA has the structure of a recurrent neural network, Tensorflow [3] can backpropagating across it. In the second version, we make use of the derived backpropagation rule across *rl*. This reinforcement of the backpropagation further enhances the accuracy of version 1. Detailed mathematical derivations of the back propagation rule appears in the Appendix.

5.3 A Trainable Sparse Code Predictor

To build an end-to-end trainable model, the first task is to have a trainable sparse code predictor. Sparse coding is the task of recovering input sparse vectors using a linear combination of over complete basis vectors. Sparse coding has been used in the computer vision based object recognition [48], [42], [55], [103], [105]. In this section, we introduce an efficient method, called LISTA [33], for acquiring accurate approximations of optimal sparse codes.

For a given input vector $x \in \mathbb{R}^m$, the problem of sparse coding or compressed sensing is to find the optimal sparse code $a \in \mathbb{R}^n$ that minimizes a cost function that combines the reconstruction loss and an L_1 sparsity penalty on the code:

$$E\left(x, a, D\right) = \frac{1}{2}\|x - Da\|_2^2 + \beta\|a\|_1 \tag{5.1}$$

where D is the sensing matrix (also known as dictionary matrix) whose columns are normalized basis vectors, β is a coefficient controlling the sparsity penalty. For a given x and D, the optimal code is defined as $\hat{a} = \arg\min_a E\left(x, a, D\right)$. Usually, the dictionary matrix D is optimized by minimizing $E\left(x, a, D\right)$ over training set using the stochastic gradient descent method. A Gabor-like filter is often obtained after training the above system, which is able to cover the space of positions, frequencies, and orientations.

To design a trainable encoder that is able to approximate sparse codes, a desired characteristic of such encoders is that they should be continuous and almost everywhere differentiable with respect to their parameters and inputs. Differentiability in regard to the parameters can guarantee that gradient based learning methods are feasible for training. Differentiability in regard to the input enables that gradients can be back propagated to the input. The central idea is to develop a non-linear and feed-forward architecture with a definite depth that is trainable to approximate an optimal sparse code. The architecture of the encoder is denoted as $a = f\left(x, W\right)$, where W represents all the trainable parameters of the encoder. During training, the stochastic gradient descent method is performed to minimize a loss function $\mathcal{L}(W)$, which measures the squared error between the predicted code and the optimal code averaged over a training set $\left(x^1, \ldots, x^P\right)$:

$$\mathcal{L}(W) = \frac{1}{P}\sum_{p=0}^{p-1} L\left(W, x^p\right) \tag{5.2}$$

$$L\left(W, x^p\right) = \frac{1}{2}\|\hat{a}^p - f\left(W, x^p\right)\|^2 \tag{5.3}$$

where $\hat{a}^p = \arg\min_a E\left(x^p, a, D\right)$ is the optimal code for instance x^p. The training method follows the stochastic gradient descent:

$$W\left(n+1\right) = W\left(n\right) - \eta\left(j\right)\frac{dL\left(W, x^{(n \mod P)}\right)}{dW} \tag{5.4}$$

where $\eta\left(n\right)$ equals to $1/n$ to guarantee convergence.

As an effective and effecient solution to sparse coding, [33] proposed a Learned Iterative Shrinkage-Thresholding Algorithm (LISTA). Figure 5.2 presents the system overview of LISTA. Top picture shows the flow chart of the ISTA sparse coding method. The optimal sparse code is obtained by iterating: $a\left(k+1\right) = h_\alpha\left(W_e x - Sa\left(k\right)\right)$, with $a(0) = 0$, where x is the input, h_α is a coordinate wise shrinking function with threshold α, W_e is the

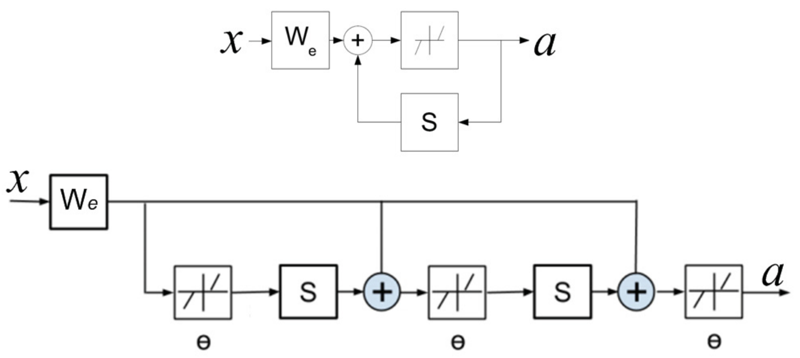


Figure 5.2: ISTA and LISTA sparse code predictor

transpose of the dictionary matrix D, and S is D^TD. Bottom picture shows the sparse encoder Learned ISTA can be seen as a time unfolded version of the ISTA method, here with 3 iterative units. The matrices W_e and S, are trained to minimize the estimation error to the optimal sparse code. The method allows us to impose restrictions on S in order to further reduce the computational complexity (e.g. keeping many terms at 0, or using a low-rank factorized form). Parameters $W = (W_e, S, \theta)$ are iteratively optimized over a set of training samples. The gradient $dL(W, x^p)/dW$ is computed during the back-propagation phase. The structure can be viewed as a time unfolded recurrent neural network (RNN), where S is shared among the layers of the RNN. We implement the trainable sparse code predictor using the deep learning library, Tensorflow [3], on which the CNN based regressor is also implemented.

In the first experiment, we would like to verify the implemented sparse code predictor by training a predictor and testing it for sparse coding task. We apply the cell location encoding scheme-2 introduced in section 4.4.1 on the dataset AMIDA-2016 [1]. Here, we choose cell location encoding scheme-2, since it requires a small sensing matrix and saves memory compared to the cell location encoding scheme 1. In this verification experiment, we use only two axes to encode the location of cells in an image. The size of each image is 200-by-200 pixels, so the length of a single sparse code encoded by each axis is $\sqrt{2} * 200 = 283$. For every sparse code, encoding scheme-2 produces a dense code (or called observation vector), which is the result of multiplication between the sparse code and the sensing matrix. Then, each sparse code and its corresponding dense code consists of the training pair for training the sparse code predictor. Given 58,700 training images of the AMIDA-2016 dataset, we will have 58,700 pairs of sparse and dense codes, on which we train the sparse code predictor. Figure 5.4 presents the training loss after training the sparse code predictor 14,400 iterations. The training process is controlled to output the training loss every 20 iterations, so the loss curve is 720 in length as shown in Figure 5.4. For the purpose of better visualization, we concatenate the two 283-dimention sparse codes (each of them corresponds to one axis) together, they become a 566-dimention sparse code as shown in each row of Figure 5.3. The first row of Figure 5.3 depicts the ground-truth sparse code after using cell location

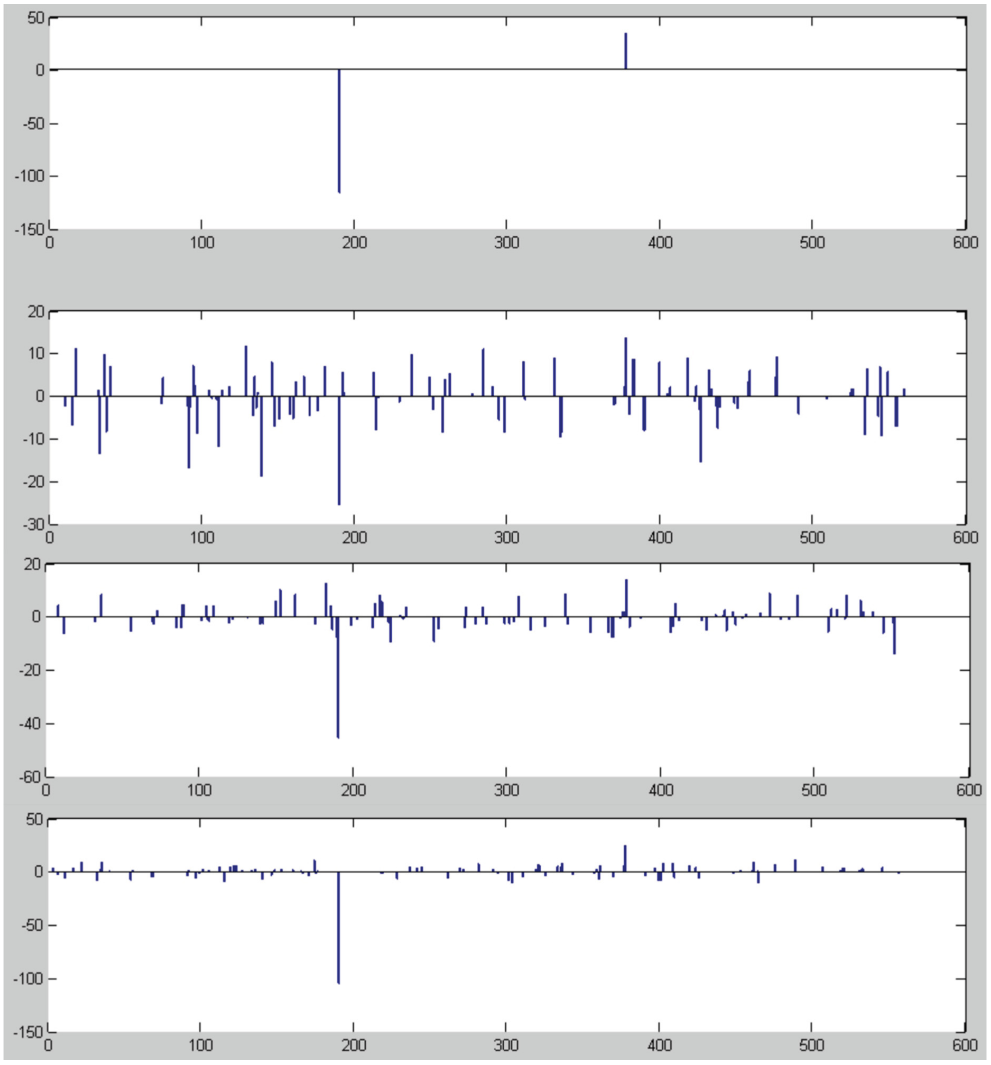


Figure 5.3: An illustration of sparse code prediction by LISTA. The 1st row shows the ground truth code. The 2nd 3rd and 4th rows present the predicted sparse by LISTA model trained after 3000, 7800 and 14400 training iterations.

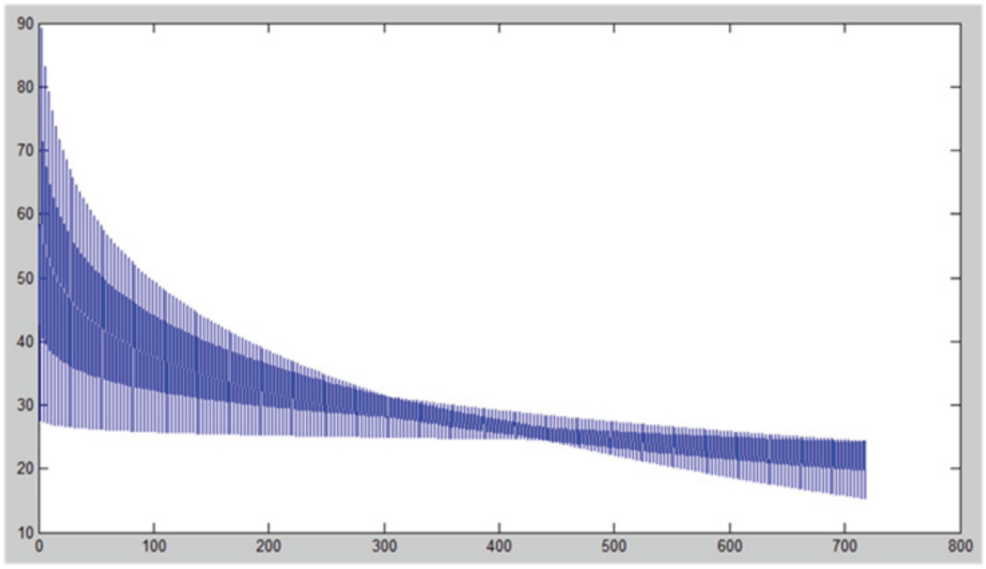


Figure 5.4: The training loss (Y-axis) after training the sparse code predictor 14,400 iterations (X-axis).

encoding scheme-2. The 2nd 3rd and 4th rows show the predicted sparse by the LISTA model trained after 3000, 7800 and 14400 training iterations.

As shown in the first row, the ground-truth code contains two non-zero elements, corresponding to one cell observed by two axes. The horizontal position of each non-zero element describes the relative positional shift of a cell along an axis. The vertical magnitude of each non-zero element indicates the perpendicular distance from a cell to an axis. The positive or negative sign represents that the cell appears on which side (right or left) of the axis line. The first information that Figure 5.3 gives us is that fewer noisy elements are predicted with as training progresses. In the sparse code predicted by the LISTA model trained after 3000 iterations (i.e. the second row), the noisy elements are hard to distinguish from the true elements. As the training process continues, the two true elements in the predicted sparse code becomes more pronounced compared to the noisy elements. This phenomenon can also be observed in the training loss as shown in Figure 5.4, where the training loss decreases fast at beginning and keeps going down with a slower convergence at later iterations. Furthermore, the vertical magnitudes of true elements become closer to their ground-truth magnitudes as the training progresses.

5.4 End-to-end trainable model: version 1

5.4.1 The Network Architecture and Loss Function

Figure 5.5 illustrates the detailed structure of the our first version of the end-to-end trainable model. During the forward pass, for each image, the *ol* predict a mL-length dense code $\hat{x} = [\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_L]^T$, from which the reconstruction layers recover the sparse code $\hat{a}$. The *ol* are built as a CNN based regressor, where a combination of deep convolutional layers with fully connected layers constructs image representations that are particularly effective for regression from raw pixels to label vectors. We utilize different CNN architectures as the core machinery for predicting the dense code. The *ol* are responsible for mapping each microscopy image to a dense code x. This fixed length dense code carries L sets of complete representation regarding the positions of target cells. In this framework, we employ L_1 norm loss to measure the loss of sparse recovery algorithm, and L_2 norm loss to measure the loss of the CNN based regression. During the backward pass, a weighted total loss combining on the L_1 loss and L_2 loss will be back propagated to adjust the parameters of observation layers and reconstruction layers. The weighted overall loss is designed as

$$loss = \frac{1}{2}\|\hat{x} - x\|_2^2 + \lambda\|\hat{a} - a\|_1 \tag{5.5}$$

where λ balances between dense code errors and sparse code errors. To optimize the weights in observation layers and reconstruction layers jointly, we train the whole model according to the overall loss (5.5) using gradient descent during backpropagation. Supervision information are from x and a, both of them carry pixel level information about true cell locations. We implemented this end-to-end trainable model using Tensorflow [3].

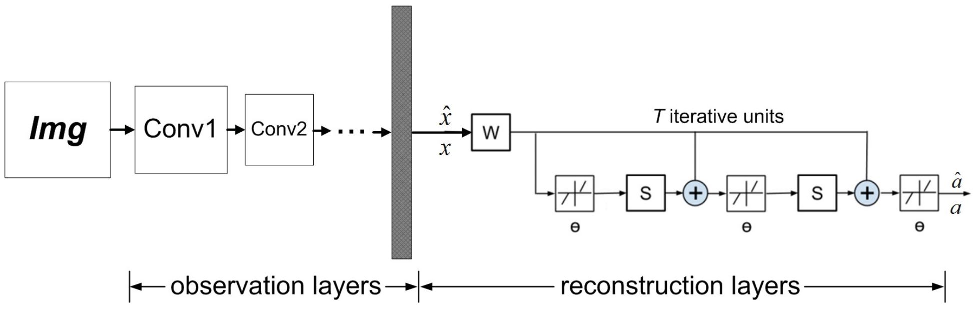


Figure 5.5: The network architecture of the End-to-End training model.

5.4.2 Model Optimization

Figure 5.6 presents the loss curves during training CNN, LISTA, and F1 scores on validation set. Here, the CNN is fine tuned from previous ResNet CNNCS, with $\lambda = 1.5$. A total of 14,400 iterations were used (display training loss every 20 iterations). LISTA depth T=30 was used. After every 2,880 training iterations, we evaluated the trained model on the validation set of AMIDA-2016 dataset to obtain the Precision, Recall and F1 score. The F1 scores have been plotted in Figure 5.6. The corresponding Precision and Recall values are shown in Table 5.1.

As Figure 5.6 shows, the training loss of LISTA starts from a relatively high value since it is trained from scratch, and then keeps decreasing with the training iterations. The fine tuning loss of CNN shows a more gradual decrease. Only fully connected layers of the CNN

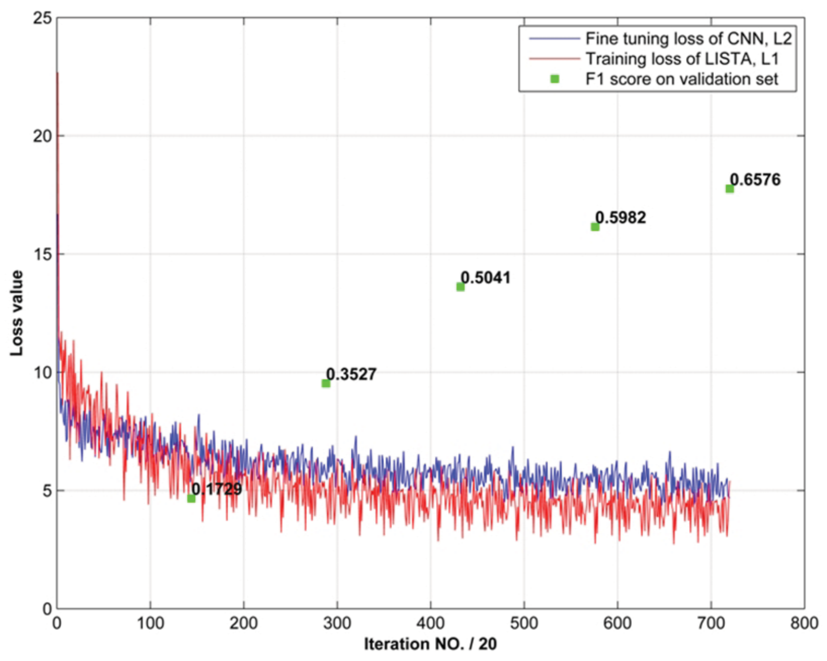


Figure 5.6: Training loss of the end-to-end trainable model

model was fine tuned. The F1 scores on the validation shows continued improvements over training iterations.

Table 5.1: Cell detection performance with respect to training iteration on validation set of AMIDA-2016.

Training iteration	Precision	Recall	F_1-score
2,880 iterations	0.1641	0.1828	0.1729
5,760 iterations	0.3472	0.3583	0.3527
8,640 iterations	0.4605	0.5567	0.5041
11,520 iterations	0.5459	0.6616	0.5982
14,400 iterations	0.5936	0.7370	0.6576

To better understand the model, we conducted several experiments, where the influence of important hyper parameters to cell detection performance was explored. After exploring the hyper parameters one by one, random search was applied to find the optimal setting of the hyper parameters. The 1st hyper parameter is λ in (5.5). The Precision, Recall and F_1-scores under different values of λ are given in Table 5.2.

λ controls the proportion of sparse loss in the total loss. Our experiment shows that it is better to set λ larger than 1.0, since it will not lead to loss of supervision information provided by error of LISTA. However, we notice that both “Precision” and “Recall” decrease when λ is increased to 2.0. A probable explanation is that a high “Precision” will result in a poor optimization of CNN. Since the CNN is fine-tuned from a pre-trained CNNCS model, it does not show a sharp decline in the F1-scores, as λ decreases.

The 2nd hyper parameter is the number of LISTA’s iterative units, i.e. T. From our

Table 5.2: Cell detection performance with respect to λ on validation set of AMIDA-2016.

λ	Precision	Recall	F_1-score
λ=0.3	0.3772	0.4081	0.3920
λ=0.5	0.4399	0.4536	0.4466
λ=1.0	0.5258	0.5826	0.5527
λ=1.5	0.5922	0.7284	0.6533
λ=1.7	0.5943	0.7304	0.6554
λ=2.0	0.5861	0.7195	0.6460

previous experiments, we know that a higher T can help LISTA to recover sparse codes better with a lower reconstruction loss. The Precision, Recall and F_1-score under different values of T are given in Table 5.3. We can see that the F_1-score increases with the increase of T. But after $T = 30$, the benefit from increasing T becomes smaller, probably due to a vanishing gradient problem in the RNN.

Table 5.3: Cell detection performance with respect to T on validation set of AMIDA-2016.

T	Precision	Recall	F_1-score
T=5	0.5103	0.6792	0.5828
T=20	0.5764	0.7215	0.6408
T=30	0.5936	0.7370	0.6576
T=45	0.5983	0.7370	0.6604

The 3rd hyper parameter is the bandwidth parameter of mean-shift clustering. Since we perform mean-shift clustering on all the detected candidate points by observation axes to obtain the final detection. This parameter controls the tightness of the clustering. The F_1 scores under different values of bandwidth are given in Table 5.4. The Precision, Recall under different values of Bandwidth are also given in Table 5.4.

Table 5.4: Cell detection performance with respect to clustering bandwidth on validation set of AMIDA-2016.

Bandwidth	Precision	Recall	F_1-score
Bandwidth=25 pixels	0.1789	0.4208	0.2511
Bandwidth=35 pixels	0.3420	0.5821	0.4309
Bandwidth=45 pixels	0.5972	0.7263	0.6555
Bandwidth=55 pixels	0.4852	0.5609	0.5203
Bandwidth=65 pixels	0.3127	0.2586	0.2831

In general, the bandwidth parameter has a strong influence on both the “Precision” and “Recall”. A higher Bandwidth will result merge more points into the same clusters, reducing “Recall”. This is undesirable when we have multiple target cells in an image. On the other hand, setting a small bandwidth parameter will decrease “Precision”.

The 4th hyper parameter is the number of observation axes, i.e. L, which is another important parameter. To explore the performance benefit limit of doing ensemble averaging with observation axes, L is set as high as 100. And corresponding Precision, Recall and

F_1-score under different values of L are given in Table 5.6. Three representative examples of detections are illustrated in Figure 5.13. While a large value of L is better for ensemble averaging, it also introduces more error for CNN prediction, becuase CNN needs to predict a larger vector. This behaviour is reflected in in Table 5.6

Table 5.5: Cell detection performance with respect to L on validation set of AMIDA-2016.

L	Precision	Recall	F_1-score
L=2	0.5619	0.5587	0.5603
L=15	0.5790	0.6172	0.5975
L=25	0.5936	0.7370	0.6576
L=35	0.6175	0.7189	0.6644
L=100	0.5728	0.6847	0.6238

Table 5.6: Cell detection performance with respect to M on validation set of AMIDA-2016.

M	Precision	Recall	F_1-score
M=90	0.5619	0.5587	0.5603
M=100	0.5790	0.6172	0.5975
M=110	0.6175	0.7189	0.6644
M=120	0.5936	0.7370	0.6576

Based on the above experiments, we acquire some basic understanding of the influence of each important hyper parameter on the overall detection accuracy. Then we perform random search by choosing a narrow ranges for the hyper parameters, and tune them jointly. After that, we obtain a group of hyper parameters and cooresponding cell detection result as shown in Table 5.7.

Table 5.7: Hyperparameters tuning by random search on validation set of AMIDA 2016 dataset.

M	L	T	Bandwidth	λ	Precision	Recall	F_1-score
115	35	31	43	1.591	0.5317	0.6689	0.5924
109	39	38	44	1.465	0.6082	0.7384	0.6670
118	41	40	42	1.517	0.6011	0.7374	0.6623
120	36	42	46	1.788	0.5536	0.7348	0.6315
102	32	35	43	1.474	0.5791	0.5613	0.5701
112	40	41	48	2.192	0.5589	0.6076	0.5823
122	37	33	51	1.571	0.6442	0.6949	0.6686
115	31	48	47	1.592	0.5462	0.5524	0.5493
116	42	36	42	1.963	0.5820	0.7403	0.6517
123	40	33	45	1.509	0.5570	0.6679	0.6075
106	38	39	48	1.415	0.6373	0.6104	0.6236
115	33	46	41	1.142	0.4973	0.7367	0.5937

Table 5.8: Cell detection performance of the end-to-end trainable model version 1.

Dataset	Precision	Recall	F1-score
AMIDA-2016 (validation)	0.6442	0.6949	0.6686
AMIDA-2013 (testing)	0.5988	0.6028	0.6008
ICPR-2014 (testing)	0.6421	0.6745	0.6579
ICPR-2012 (testing)	0.8637	0.8385	0.8509

5.5 End-to-end trainable model: refined version

After building the initial end-to-end trainable model, I am still interested to further refine the model. According to my experiment and also pointed out by [33], if we increase the number of iterative units, which means we build deeper reconstruction layers, we can reduce the sparse code reconstruction error. But after more experiments, we also find that using more iterative units is not always helpful. Because although from the perspective of sparse code reconstruction, deeper reconstruction layers are able to reduce reconstruction error, but from the perspective of the whole end-to-end trainable model, deeper reconstruction layers will lead us to the problem of gradient vanishing. It means that gradient tends to get smaller and smaller, even vanish, when it moves backward closer to the layers at the input end. As a result, the neurons in the bottom layers are hard to be fully optimized. Figure 5.7 illustrates the influence of vanishing gradient problem to the network training process.

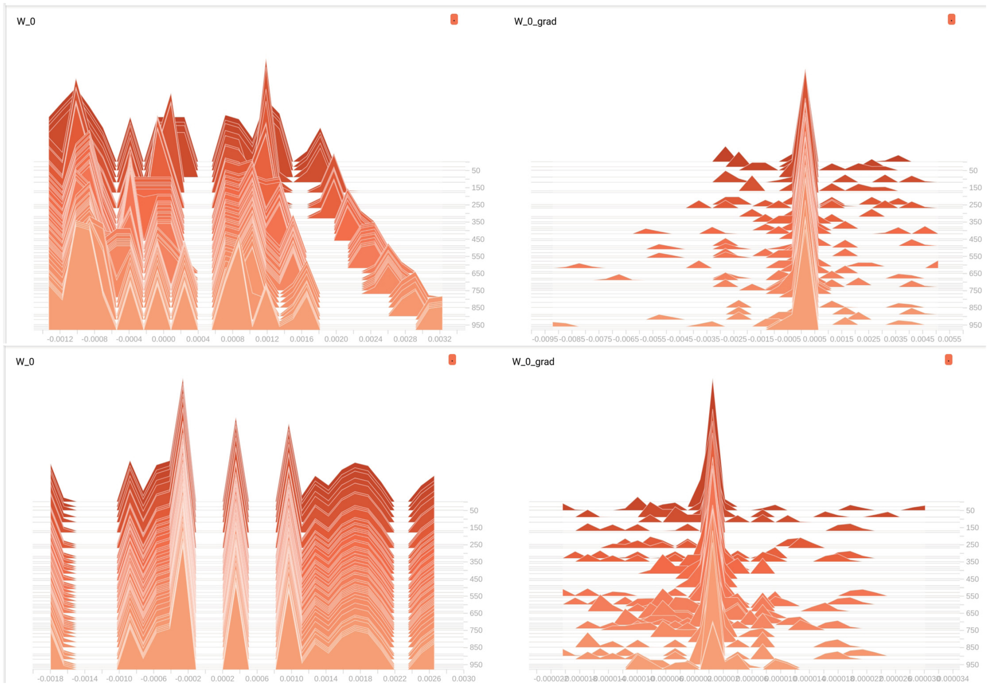


Figure 5.7: An illustration of the evolution of values in the first observation layer through a training process seen in TensorBoard, under the "histograms" tab.

Figure 5.7 visualizes weights (left) and gradients (right) of the first observation layer and how their value distribution (X-axis) change from training iteration 0 to 1000 (Y-axis).

The top panel shows the weights and the gradients when 5 iterative units are used in the reconstruction layers. The bottom panel corresponds to the weights and the gradients when 50 iterative units are used in the reconstruction layers. In both top and bottom panel, observation layers follow the architecture of AlexNet, and made up of 5 convolutional layers and 3 full-connected layers. As we can see, the change of weights is much more obvious in the top panel; in comparison, the weights in the bottom panel are not evolving. Furthermore, the gradient in the bottom panel is quite close to 0. Its major distribution is in the range between -0.000002 to 0.000002. In comparison, the gradient in the top panel is in the range between -0.005 to 0.005, i.e. 2500 times larger. Weights not evolving and extremely small gradients are the symptom of the vanishing gradient problem, which we should pay more attention when the neural network becomes deeper. In order to solve this problem, I derive my own back propagation rule for sparse coding. And the back propagation rule is independent of specific sparse coding algorithm. In other words, for LISTA [33], for DAL [90], for OMP [9], or any other sparse coding algorithm, the back propagation rule is effective. To do that, lets first go back to the sparse coding problem itself.

5.5.1 Mathematical Derivation of Back Propagation Rule

The central idea of L_1 minimization algorithm is to find an optimized sparse vector $\hat{a}$ by minimizing cost function (5.6), where D is the sensing matrix and x indicates the input dense code:

$$\frac{1}{2}\|Da - x\|_2^2 + \lambda\|a\|_1 \tag{5.6}$$

In order to update weights of CNN and sensing matrix during the backward pass, we differentiate (5.6) with respect to x and D. The details of mathematical derivation can be found in the Appendix. Here we provide the derivation results:

$$\delta x = D\left(:, p\right)\left[D^T D\left(p, p\right)\right]^{-1}\delta a\left(p\right) \tag{5.7}$$

$$\delta D\left(:, p\right) = \left(x - Da\right)\delta a\left(p\right)^T\left[D^T D\left(p, p\right)\right]^{-1} - D\left(:, p\right)\left[D^T D\left(p, p\right)\right]^{-1}\delta a\left(p\right) a\left(p\right)^T \tag{5.8}$$

where a stands for the output vector from sparse recovery, and contains the predicted location information of cells. δa, δx and δD are the partial derivatives of the overall network loss with respect to a, x, and D, respectively. D is the sensing matrix. $p = \{i : a_i \neq 0\}$ is a set of indices. $D(:, p)$ indicates the columns of matrix D, whose indices belong to the set p. $D^T D\left(p, p\right)$ indicates the principal sub matrix of $D^T D$ with column and row indices belonging to set p. Based on (5.7) and (5.8), the network is able to optimize or update x and D during the back propagation, so that the whole framework can be trained in an End-to-End fashion.

To experimentally verify these derivations (5.7) and (5.8), we perform numerical gradient check on the above partial derivatives δx and δD. During the gradient check, we first compute the numerical gradient of δx according to:

$$\delta x_i \approx \frac{||a - c_i||_2^2 - ||a - b||_2^2}{h} \quad i = 1, 2, \ldots, m. \tag{5.9}$$

where a, b and c are sparse codes corresponding to dense code x, y and $y + e_i h$, respectively. h indicates a small deviation value. e_i is a m-length one-hot vector with only one positive at its i-th position. $e_i = [0, \ldots, 1, \ldots, 0], i = 1, 2, \ldots, m$. Then, we compute the gradient according to backpropagation rule (5.7). Figure 5.9 shows that the numerical gradient (5.9) and the gradient computed according to backpropagation (5.7) are consistent with each other.

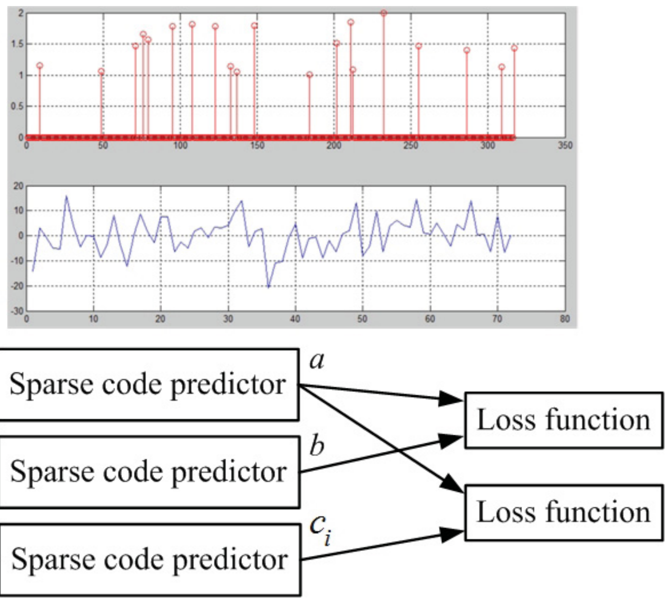


Figure 5.8: An explanation diagram about how we compute numerical gradients. The top two panels present two examples of a sparse code a and a dense code x. The bottom panel shows the relationship between several important notations.

5.5.2 Efficient Back Propagation

Our derived backpropagation rules (5.7) and (5.8) are not computationally efficient for a batch training, which is a normal practice for training a neural net. To examine why it is so, note that the partial derivatives depend on the set p, which is in turn different sparse codes a for the member of the batch. Thus, rules (5.7) and (5.8) require matrix inversion for each member of the batch, rendering the rule impractical in batch stochastic gradient descent. This observation forces us to develop an approximate gradient computation, which would be batch friendly.

Our first observation is that in equations (5.7) and (5.8), $D^T D$ is close to a diagonal matrix when D is random Gaussian. Actually, any incoherent matrix D will induce this property to $D^T D$. An example of $D^T D$ is visualized in Figure 5.10. Second, if we examine any rows of $D^T D$, we find that the values of diagonal elements are dominant compared to the values of elements in other positions as shown in Figure 5.11. So we can approximate $D^T D$ by its diagonal matrix $diag(D^T D)$, which retains the original values of $D^T D$ at the diagonal. In other words, we approximate $\left[D^T D\left(p, p\right)\right]^{-1}$ by $\left[diag\left(D^T D\right)\left(p, p\right)\right]^{-1}$. Then

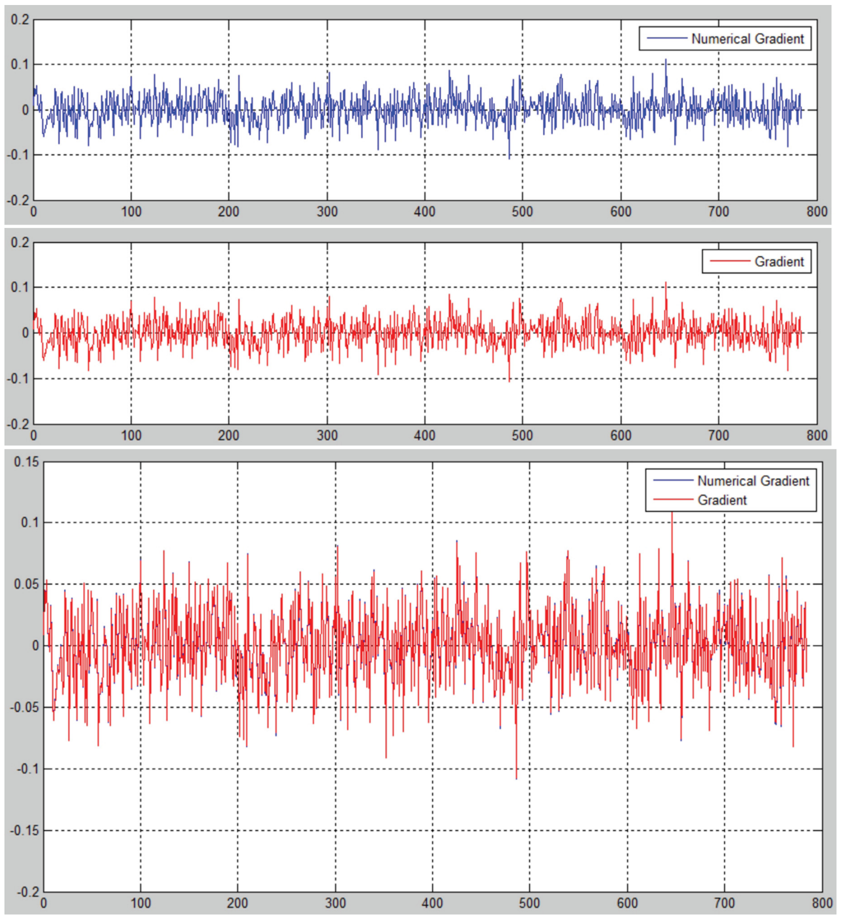


Figure 5.9: Gradient check between the gradient δx computed according to equation (5.7) and numerical gradient. They are simultaneously plotted in the third panel, where two curves almost overlap with each other, with only small blue tips can be seen above some red peaks.

because $diag(D^TD)$ is a diagonal matrix, so its inversion matrix become very simple, which is the inversion of its diagonal elements.

$$\left[diag\left(D^TD\right)(p,p)\right]^{-1} = \left[diag\left(D^TD\right)\right]^{-1}(p,p) \tag{5.10}$$

Equation (5.10) implies that for the entire batch, we need to invert the diagonal matrix once that is also computationally trivial.

However, numerical instability may still arise during inversion of the diagonal matrix. Thus, we further note that the diagonal elements of D^TD should be very similar to each other considering D as a random Gaussian matrix. This observation is also clear from Figure 5.11. So, practically we can even ignore the matrix inversion altogether and arrive at a numerically stable and batch friendly backpropagation rule for δx as follows:

$$\delta x = D\left(:,p\right)\delta a\left(p\right). \tag{5.11}$$

Similarly, for δD, a practical backpropagation rule would be:

$$\delta D\left(:,p\right) = \left(x - Da\right)\delta a\left(p\right)^T - D\left(:,p\right)\delta a\left(p\right)a\left(p\right)^T \tag{5.12}$$

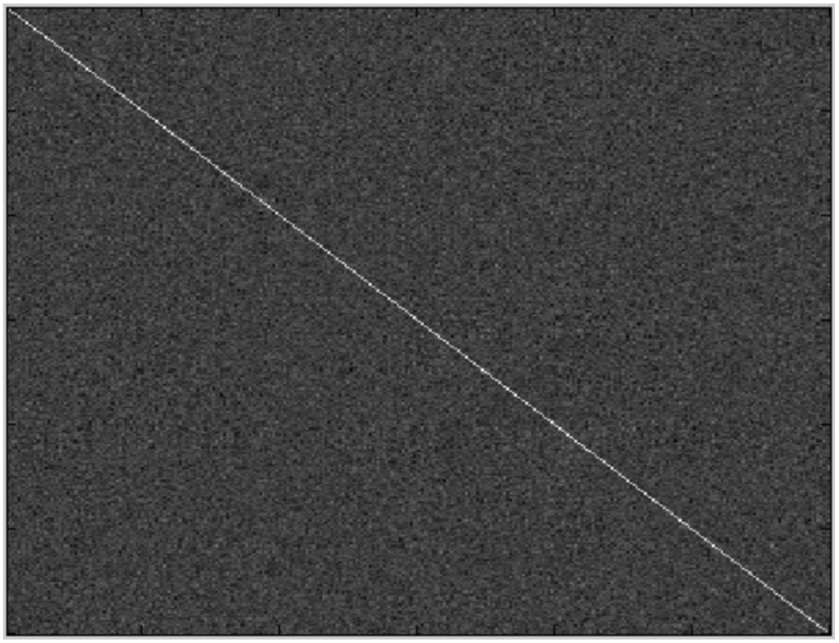


Figure 5.10: An illustration of the diagonal matrix D^TD.

5.5.3 Experimental evidence for end-to-end training

In order to find out, whether the end-to-end training is indeed helpful or not, we carry out experiments with the two versions of the end-to-end models and compare these results with CNNCS. However, while CNNCS uses DAL algorithm for L_1 recovery, while our end-to-end model uses LISTA for the same. Thus, to make a fair comparison with the non-end-to-end

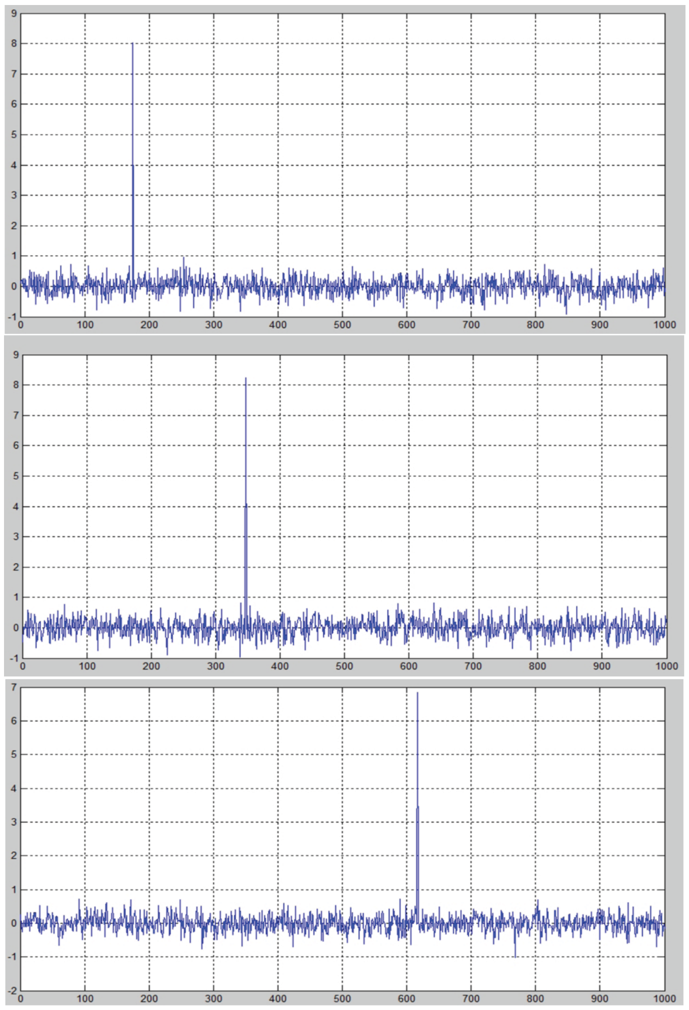


Figure 5.11: Three randomly chosen rows of the diagonal matrix $D^T D$.

version, we use ISTA in place of DAL. So that at the component level, the CNNCS model and the end-to-end trainable model are equivalent. Then, we train the CNNCS model again using the same settings as discussed in chapter 4 and obtain its performance on four evaluation datasets. The performance comparisons are shown in Table 5.9-Table 5.12, which summarize the results of the original CNNCS model, the CNN-ISTA model, the end-to-end trainable model version 1 and the end-to-end trainable model version 2 on the datasets: AMIDA-2016, AMIDA-2013, ICPR-2014, and ICPR-2012. Here, the CNN-ISTA model is the modified CNNCS for comparison. It can be seen that CNN-ISTA shows a similar level of F_1 score compared with CNNCS. But the two versions of the end-to-end trainable model outperform CNN-ISTA with significant improvements about 2%-15% in terms of their F_1 scores on four datasets. These comparions indicate that the performance improvement is owing to the end-to-end training.

Table 5.9: Performance of end-to-end trainable models on validation set AMIDA-2016 dataset.

Method	Precision	Recall	F1-score
CNNCS	0.5646	0.7240	0.6344
CNN-ISTA	0.5478	0.7314	0.6264
End-to-end trainable v1	0.5936	0.7370	0.6576
End-to-end trainable v2	0.6423	0.7352	0.6856
Faster-rcnn	0.4961	0.7728	0.6043

Table 5.10: Cell detection performance of four models on testing set of AMIDA-2013 dataset.

Method	Precision	Recall	F1-score
CNNCS	0.3588	0.5529	0.4352
CNN-ISTA	0.3952	0.5879	0.4727
End-to-end trainable v1	0.5988	0.6028	0.6008
End-to-end trainable v2	0.6137	0.6541	0.6332
Faster-rcnn	0.4875	0.7319	0.5852

Table 5.11: Performance of end-to-end trainable models on testing set of ICPR-2014 dataset.

Method	Precision	Recall	F1-score
CNNCS	0.6157	0.6510	0.6329
CNN-ISTA	0.6210	0.6527	0.6365
End-to-end trainable v1	0.6421	0.6745	0.6579
End-to-end trainable v2	0.6384	0.7056	0.6703
Faster-rcnn	0.5351	0.7520	0.6253

In addition, I also compare the method with state-of-the-art detection method faster-rcnn [74]. I follow the implementation of faster-rcnn [1]. I download the faster-rcnn model, which is an ImageNet-pre-trained VGG-16 net. Then fine tune the faster-rcnn model on the cell datasets. To build training data,I generate a bbox in size of 50*50 (the biggest size of mitosis) around every true target cell. As you can see, faster-rcnn gives F1-scores that

[1] `https://github.com/ShaoqingRen/faster_rcnn`

Table 5.12: Cell detection performance of four models on testing set of ICPR-2012 dataset.

Method	Precision	Recall	F1-score
CNNCS	0.872	0.805	0.837
CNN-ISTA	0.853	0.791	0.821
End-to-end trainable v1	0.8637	0.8385	0.8509
End-to-end trainable v2	0.8793	0.8454	0.8620
Faster-rcnn	0.5334	0.8152	0.6440

are slightly lower than the original CNNCS method, but much lower than the end-to-end trainable models. Figure 5.12 visualizes the detection results given by faster-rcnn. Yellow box is the prediction by faster-rcnn. Green box is the ground truth. For each prediction, there is a confidence score attached.

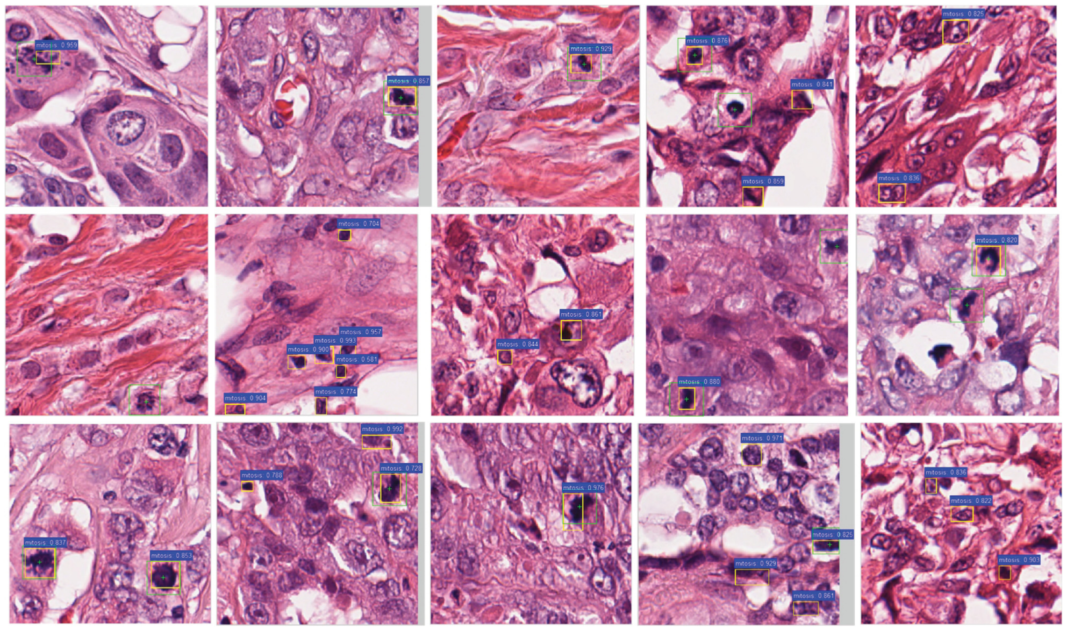


Figure 5.12: Cell detection results given by Faster-rcnn.

5.5.4 Cell detection result by end-to-end trainable model

Figure 5.13 visualizes three examples of cell detection result by End-to-End framework, and the ground-truth (top) and predicted (bottom) sparse code comparison. Each green cross indicates a cell location predicted by an observation axis; yellow cross indicates the ground-truth position of the target cell. The size of each image is 200-by-200 pixels, so the length of a single sparse code encoded by each axis is $\sqrt{2} * 200 = 283$. Now we use 25 observation axes. For the purpose of a better visualization, we concatenate 25 sparse codes (each of them is in 283-dimention and corresponds to one axis line) together, they become a 7075-dimention long sparse code, which is shown in the right panels of Figure 5.13. In the left panel we show two example detections.

In addition to the successful detection, Figure 5.14 also provides two representative failure detection examples. For the proposed cell detection method, the biggest challenge

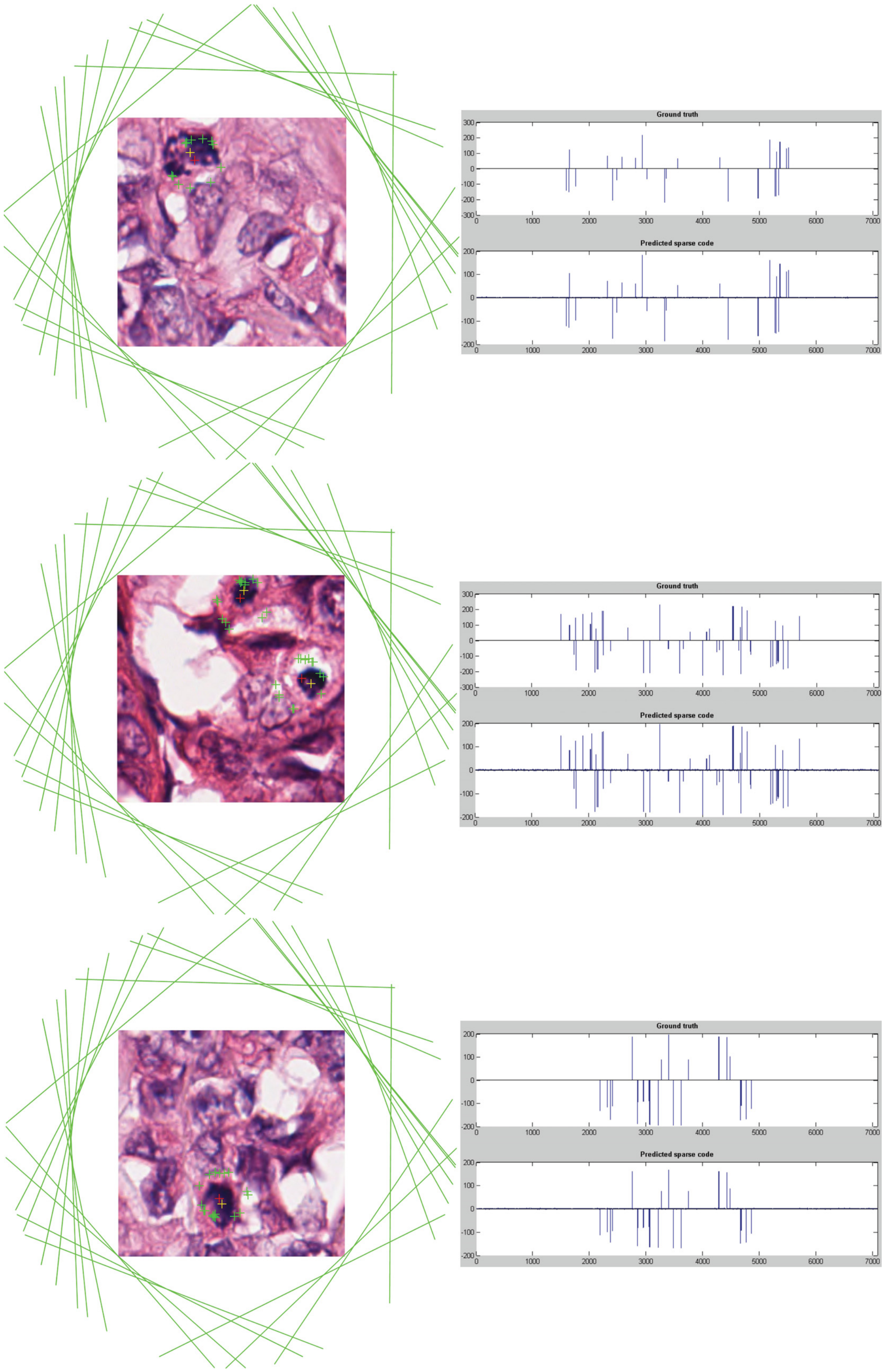
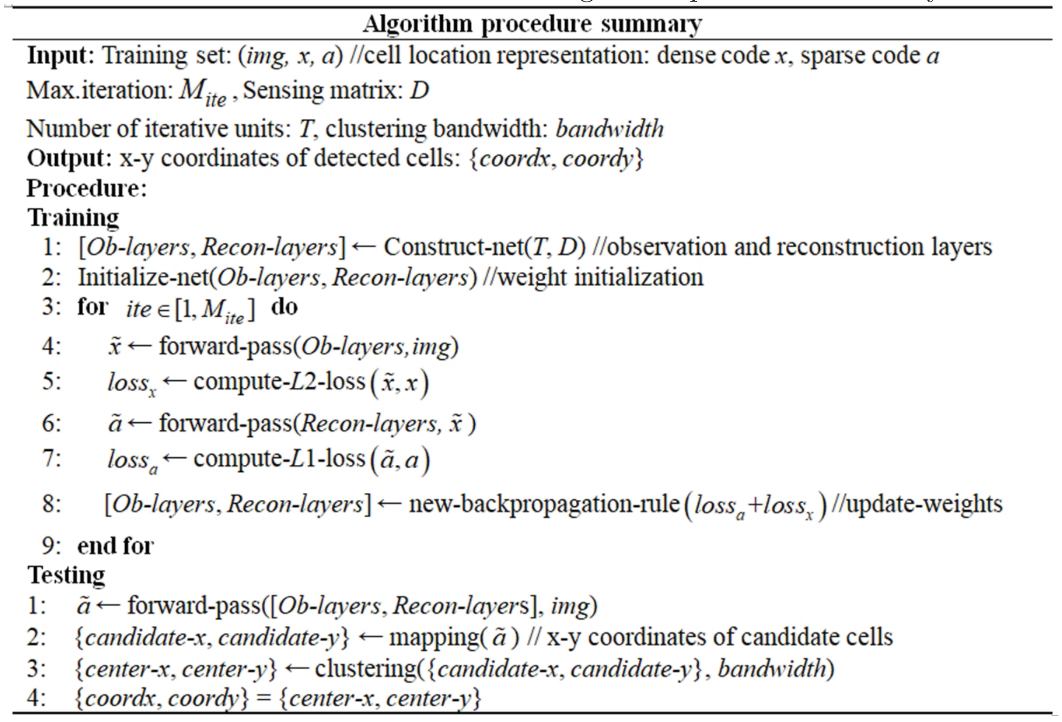


Figure 5.13: Left figures show the examples of cell detection results. Right figures present the ground-truth (top) and predicted (bottom) sparse code. Yellow cross indicates the ground-truth position of target cells. Green cross indicates cell position predicted by an observation axis. Red cross indicates the final detected cell position, which is the clustering center of all green crosses.

Table 5.13: Cell detection performance of the end-to-end trainable model version 2.

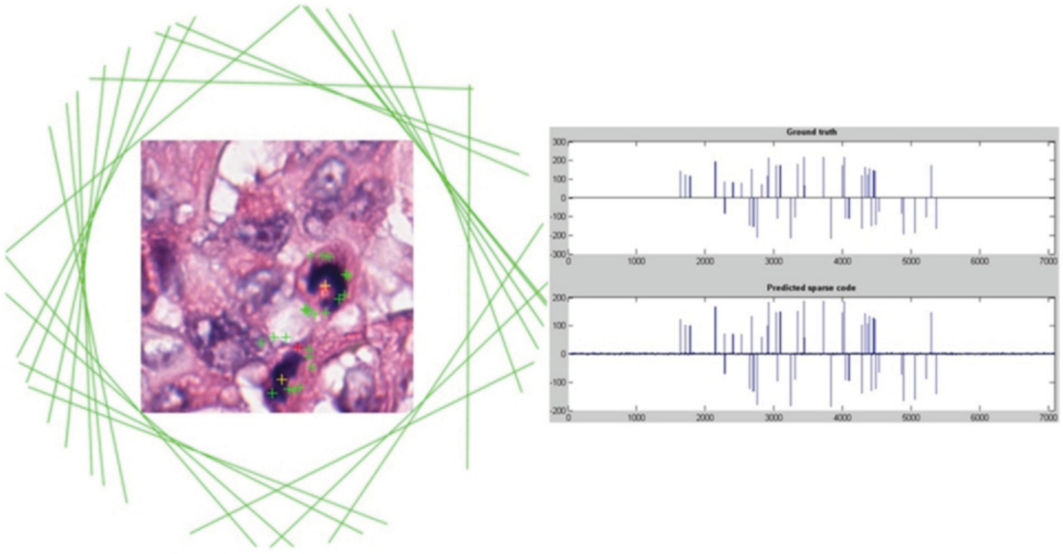
Dataset	Precision	Recall	F1-score
AMIDA-2016 (validation)	0.6423	0.7352	0.6856
AMIDA-2013 (testing)	0.6137	0.6541	0.6332
ICPR-2014 (testing)	0.6384	0.7056	0.6703
ICPR-2012 (testing)	0.8793	0.8454	0.8620

Table 5.14: End-to-end trainable algorithm procedure summary.



```
Algorithm procedure summary
Input: Training set: (img, x, a) //cell location representation: dense code x, sparse code a
Max.iteration: $M_{ite}$, Sensing matrix: D
Number of iterative units: T, clustering bandwidth: bandwidth
Output: x-y coordinates of detected cells: {coordx, coordy}
Procedure:
Training
  1: [Ob-layers, Recon-layers] ← Construct-net(T, D) //observation and reconstruction layers
  2: Initialize-net(Ob-layers, Recon-layers) //weight initialization
  3: for $ite \in [1, M_{ite}]$ do
  4:     $\tilde{x}$ ← forward-pass(Ob-layers, img)
  5:     $loss_x$ ← compute-L2-loss$(\tilde{x}, x)$
  6:     $\tilde{a}$ ← forward-pass(Recon-layers, $\tilde{x}$)
  7:     $loss_a$ ← compute-L1-loss$(\tilde{a}, a)$
  8:     [Ob-layers, Recon-layers] ← new-backpropagation-rule$(loss_a + loss_x)$ //update-weights
  9: end for
Testing
1:  $\tilde{a}$ ← forward-pass([Ob-layers, Recon-layers], img)
2:  {candidate-x, candidate-y} ← mapping($\tilde{a}$) // x-y coordinates of candidate cells
3:  {center-x, center-y} ← clustering({candidate-x, candidate-y}, bandwidth)
4:  {coordx, coordy} = {center-x, center-y}
```

comes when there are multiple target cells in an image, especially when these target cells are spatially close. If we use more observation lines e.g. 25 and 100 lines are used respectively. There are many candidate points (as shown in green color) detected by each observation axis line. Under the above circumstance, these candidate points are very likely to be mixed together. Thus it is hard to find the acceptable detection results by clustering on them. Table 5.14 provides an algorithm procedure summary.



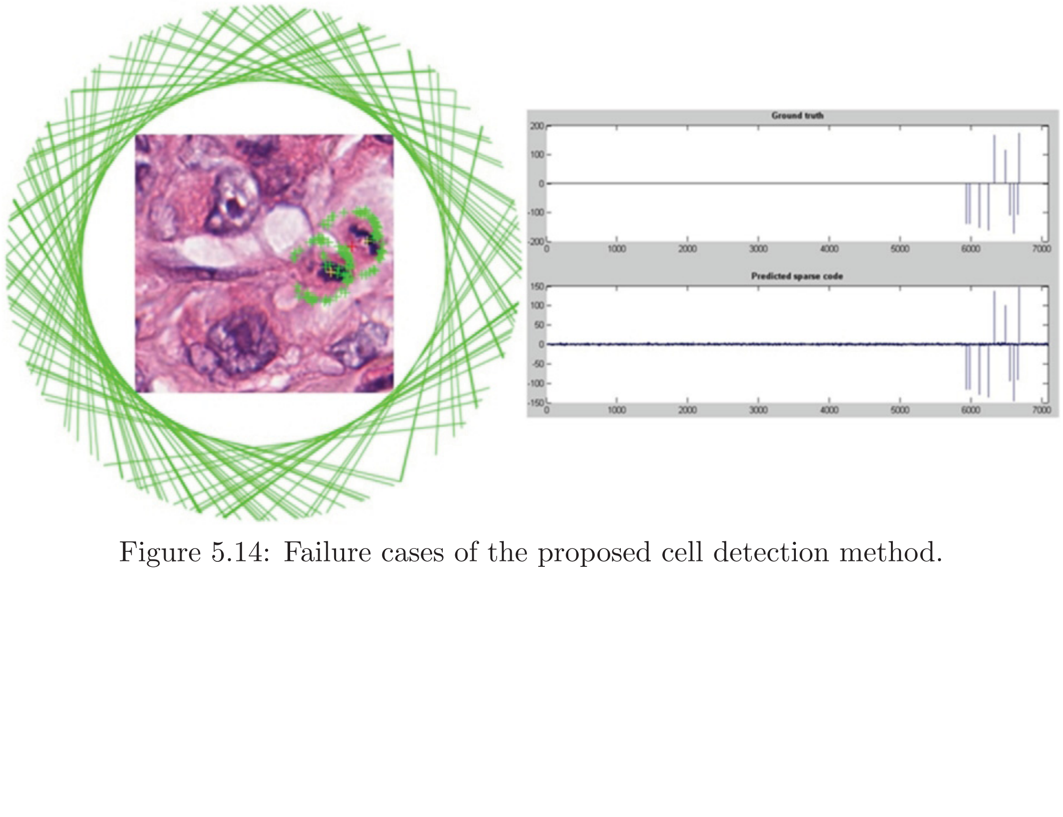


Figure 5.14: Failure cases of the proposed cell detection method.

Chapter 6

Conclusion and Future work

First, we present a supervised learning framework with Convolutional Neural Network (CNN) and cast the cell counting task as a regression problem, where the global cell count is taken as the annotation to supervise training. To capture the relationship between RGB cell image and its overall cell count, cell counting task is formulated as developing an end-to-end regression framework, which is more suitable for counting task instead of following the classification or detection framework. Additionally, instead of being applied for classification purpose, a convolutional neural network architecture with Euclidean loss function is used for regression. To further decrease the prediction error of counting, we utilize several popular CNN architectures (including the Deep Residual Network, AlexNet) into our regression model. To the best of our knowledge, this is the first piece of work to expand the deep residual network from classification, detection, segmentation to object counting. As the final output, the proposed approach not only estimate the global number of certain cells in an image but also produce the spatial density prediction, which is able to describe the local cell density of an image sub-region. In many clinical imaging systems, researchers have confirmed that the topographic map that illustrates the cell density distribution is a valuable tool correlated with the disease diagnose and treatment. The proposed method was compared with three state-of-the-art approaches on three cell image datasets and obtain superior performance.

Second, cell detection methods have evolved from employing hand-crafted features to deep learning-based techniques. The essential idea of these methods is that their cell classifiers or detectors are trained in the pixel space, where the locations of target cells are labeled. Furthermore, treating the cell detection/localization as a regression task is tricky, because the length of output cell centroid locations is unknown. Thus, regressing on a fixed length vector is hardly an option. Our key observation for cell detection or localization is that the output space is quite sparse: an automated system needs to label a small fraction of the total pixels as cell centroid locations. We seek a different route and propose a convolutional neural network (CNN)-based cell detection method that uses encoding of the output pixel space. We employ compressed sensing (CS)-based output encoding here. Using random projections, CS encodes the sparse, output pixel space into a dense and compressed vector of fixed dimension. Then, we use a deep convolutional neural network based regressor to predict this compressed vector from the input pixels. Furthermore, it is possible to stably recover sparse cell locations on the output pixel space from the predicted compressed vector using L_1-norm optimization. In the past, output space encoding using compressed sensing (CS) has been used in conjunction with linear and non-linear predictors. To the best of our knowledge, this is the first successful use of CNN with CS-based output space encoding. We demonstrate CS-based output encoding provides us with the opportunity to do ensemble averaging to boost detection/localization scores. We made substantial experiments on several benchmark datasets, where the proposed CNN + CS framework (referred to as CNNCS) achieved the highest or at least top-3 performance in terms of F1-score, compared with other state-of-the-art methods. The contribution of the proposed algorithm can be summarized as follows. First, this is the first attempt to combine deep learning with CS-based output encoding to solve cell detection and localization problem. Second, we try to overcome the aforementioned class imbalance issue by converting a classification problem into a regression problem, where sparse cell locations are distributed by a random projection into a fixed length vector as a target for the regression. Third, we introduce redundancies in the CS-based output encoding that are exploited by CNN to boost generalization accuracy in cell detection and localization. This redundancies also help to reduce false detections. Fourth, we demonstrate that the proposed CNNCS framework achieves competitive results compared to the state-of-the-art methods on several benchmark datasets and challenging cell detection contests.

The CNNCS framework consists of two components used in a cascaded pipeline. Any deviations prediction error from the first module will pass to the second module and affect the accuracy of the recovered sparse code predicted by the second module. To tackle this problem, on the basis of the proposed CNNCS model, I further develop an end-to-end trainable model, which makes all the parameters of CNNCS's two key components able to be optimized jointly, and avoids non-trivial pre-processing or post-processing operations, which requires significant prior knowledge about the task and the dataset. In the end-to-end trainable model, the loss value from the very end will be back propagated through the whole architecture and the parameters of both the CNN based regressor and the CS based sparse code predictor can be adjusted jointly during back propagation. Furthermore, we make detailed mathematical derivations on the partial derivative of total loss with to network parameters and perform gradient check to verify the mathematical derivations. After that, we modify the error back propagation rule for training the end-to-end model, so as to overcome the vanishing gradient problem. It is the first work that derives a back propagation rule for a sparse coding (i.e., compressed sensing) algorithm. Our back propagation rule is independent of a sparse coding/compressed sensing algorithm. Also, our experiments on benchmark datasets show that our back propagation method increases accuracy by virtue of end-to-end training.

I believe the CNNCS framework has big application potential in other point object detection problem (e.g. crowded people detection) and general object detection problem (e.g. vehicle detection with bounding boxes). The proposed method is not specific to cell (centroid) detection. Because the method doesn't care the encoded point is a centroid of a cell or a corner point of a bounding box. For object detection with bounding boxes, the prediction of a bounding box can be formulated as prediction of two diagonal points of a bounding box, then my method can be transplanted there without big algorithm level modifications. In summary, the biggest future plan is to transform the idea behind CNNCS into a more general object detection method.

Bibliography

[1] Amida-2016 mitosis detection challenge homepage. `http://tupac.tue-image.nl/`.

[2] Icpr-2014 mitosis detection challenge homepage. `https://mitos-atypia-14.grand-challenge.org/home/`.

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[4] Yousef Al-Kofahi, Wiem Lassoued, William Lee, and Badrinath Roysam. Improved automatic detection and segmentation of cell nuclei in histopathology images. *IEEE Transactions on Biomedical Engineering*, 57:841–852, 2010.

[5] Dwi Anoraganingrum. Cell segmentation with median filter and mathematical morphology operation. In *Proceedings of the 10th International Conference on Image Analysis and Processing*, ICIAP '99, pages 1043–, Washington, DC, USA, 1999. IEEE Computer Society.

[6] Carlos Arteta, Victor Lempitsky, J. Alison Noble, and Andrew Zisserman. Learning to detect cells using non-overlapping extremal regions. *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 348–356, 2012.

[7] Carlos Arteta, Victor Lempitsky, J. Alison Noble, and Andrew Zisserman. Interactive object counting. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 504–518, Cham, 2014. Springer International Publishing.

[8] Olga Barinova, Victor Lempitsky, and Pushmeet Kholi. On detection of multiple object instances using hough transforms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(9):1773–1784, September 2012.

[9] T. Tony Cai and Lie Wang. Orthogonal matching pursuit for sparse signal recovery with noise. *IEEE Transactions on Information Theory.*, 2011.

[10] Alison Noble Carlos Arteta, Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. *Neural Information Processing Systems.*, 2010.

[11] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, Jul 1997.

[12] Antoni B. Chan, Zhang-Sheng John Liang, and Nuno Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7, 2008.

[13] Hao Chen, Qi Dou, Xi Wang, Jing Qin, and Pheng-Ann Heng. Mitosis detection in breast cancer histology images via deep cascaded networks. *Proceedings of the Thirtieth Conference on Artificial Intelligence (AAAI)*, 2016.

[14] Liang Yang Si Liu Xiaochun Cao Chuan Wang, Hua Zhang. Deep people counting in extremely dense crowds. *ACM international conference on Multimedia*, 2015.

[15] Dan C. Cirean, Alessandro Giusti, Luca M. Gambardella, and Jrgen Schmidhuber. *Mitosis Detection in Breast Cancer Histology Images with Deep Neural Networks.* 2013.

[16] A. S. Coates, , E. P. Winer, , A. Goldhirsch, , R. D. Gelber, , M. Gnant, , M. Piccart-Gebhart, , B. Thrlimann, , H.-J. Senn, , , Fabrice Andr, Jos Baselga, Jonas Bergh, Herv Bonnefoi, Harold Burstein, Fatima Cardoso, Monica Castiglione-Gertsch, Alan S. Coates, Marco Colleoni, Giuseppe Curigliano, Nancy E. Davidson, Angelo Di Leo, Bent Ejlertsen, John F. Forbes, Viviana Galimberti, Richard D. Gelber, Michael Gnant, Aron Goldhirsch, Pamela Goodwin, Nadia Harbeck, Daniel F. Hayes, Jens Huober, Clifford A. Hudis, James N. Ingle, Jacek Jassem, Zefei Jiang, Per Karlsson, Monica Morrow, Roberto Orecchia, C. Kent Osborne, Ann H. Partridge, Lorena de la Pea, Martine J. Piccart-Gebhart, Kathleen I. Pritchard, Emiel J.T. Rutgers, Felix Sedlmayer, Vladimir Semiglazov, Zhi-Ming Shao, Ian Smith, Beat Thrlimann, Masakazu Toi, Andrew Tutt, Giuseppe Viale, Gunter von Minckwitz, Toru Watanabe, Timothy Whelan, Eric P. Winer, and Binghe Xu. Tailoring therapiesimproving the management of early breast cancer: St gallen international expert consensus on the primary therapy of early breast cancer 2015. *Annals of Oncology*, 26(8):1533–1546, 2015.

[17] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. *Computer Vision and Pattern Recognition*, 2005.

[18] Deepak Pathak Philipp Krahenbuhl Trevor Darrell. Constrained convolutional neural networks for weakly supervised segmentation. *ICCV*, 2015.

[19] Chaitanya Desai, Deva Ramanan, and Charless C. Fowlkes. Discriminative models for multi-class object layout. *International Journal of Computer Vision*, 95(1):1–12, Oct 2011.

[20] Xavier Descombes, Robert Minlos, and Elena Zhizhina. Object extraction using a stochastic birth-and-death dynamics in continuum. *Journal of Mathematical Imaging and Vision*, 33(3):347–359, Mar 2009.

[21] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, pages 263–286, 1995.

[22] Lan Dong, Vasu Parameswaran, Visvanathan Ramesh, and Imad Zoghlami. Fast crowd segmentation using shape indexing. pages 1–8, 01 2007.

[23] David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 2006.

[24] Nathan S. Hart Shaun P. Collin Eduardo Garza-Gisholt*, Jan M. Hemmi. A comparison of spatial analysis methods for the construction of topographic maps of retinal cell density. *PLoS One*, 2014.

[25] Justin Romberg Emmanuel Candes. Practical signal recovery from random projections. *IEEE Transactions on Signal Process*, 2005.

[26] Terence Tao Emmanuel Candesy, Justin Rombergy. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory.*, 2006.

[27] Meijering Erik. Cell segmentation: 50 years down the road [life sciences]. *IEEE Signal Process. Mag.*, 2012.

[28] Cantaloni C Eccher C Bazzanella I Aldovini D Bragantini E Morelli L Cuorvo LV Ferro A Gasperetti F Berlanda G Dalla Palma P Barbareschi M. Fasanella S, Leonardi E. Proliferative activity in human breast cancer: Ki-67 automated evaluation and the influence of different ki-67 equivalent antibodies. *Diagn Pathol.*, 2011.

[29] Luca Fiaschi, Ullrich Köthe, Rahul Nair, and Fred A. Hamprecht. Learning to count with regression forest and structured labels. *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 2685–2688, 2012.

[30] G. Flaccavento, V. Lempitsky, I. Pope, P. R. Barber, A. Zisserman, J. A. Noble, and B. Vojnovic. Learning to count cells: applications to lens-free imaging of large fields. In *Microscopic Image Analysis with Applications in Biology*, 2011.

[31] Ross Girshick. Fast r-cnn. *International Conference on Computer Vision*, 2015.

[32] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *Computer Vision and Pattern Recognition*, 2014.

[33] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 399–406, USA, 2010. Omnipress.

[34] Metin N. Gurcan, Senior Member, Laura E. Boucheron, Ali Can, Anant Madabhushi, Senior Member, Nasir M. Rajpoot, Bulent Yener, and Senior Member. Histopathological image analysis: A review. *IEEE Reviews in Biomedical Engineering*, pages 147–171, 2009.

[35] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Computer Vision and Pattern Recognition.*, 2015.

[37] Daniel Hsu, Sham M. Kakade, John Langford, and Tong Zhang. Multi-label prediction via compressed sensing. *arXiv:0902.1284v2 [cs.LG]*, 2009.

[38] Daniel J. Hsu, Sham M. Kakade, John Langford, and Tong Zhang. Multi-label prediction via compressed sensing. part of: Advances in neural information processing systems. *Neural Information Processing Systems (NIPS)*, 2009.

[39] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org, 2015.

[40] Humayun Irshad. Automated mitosis detection in histopathology using morphological and multi-channel statistics features. *Journal of Pathology Informatics*, 2013.

[41] Humayun Irshad. Automated mitosis detection in histopathology using morphological and multi-channels statistics features. *Journal of Pathology Informatics*, 4, 05/2013 2013.

[42] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, pages 2146–2153. IEEE, 2009.

[43] Arnaud Joly. Exploiting random projections and sparsity with random forests and gradient boosting methods. *arXiv:1704.08067*, 2016.

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

[45] Y.W Tsang I.A. Cree D.R.J. Snead K. Sirinukunwattana, S.E.A. Raza and N.M. Rajpoot. Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images. *IEEE Transactions on Medical Imaging.*, 2016.

[46] Y.W Tsang I.A. Cree D.R.J. Snead K. Sirinukunwattana, S.E.A. Raza and N.M. Rajpoot. Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images. *IEEE Transactions on Medical Imaging.*, 2016.

[47] Ashish Kapoor, Raajay Viswanathan, and Prateek Jain. Multilabel classification using bayesian compressed sensing. *Advances in Neural Information Processing Systems*, 2012.

[48] Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. Fast inference in sparse coding algorithms with applications to object recognition. *CoRR*, abs/1010.3467, 2010.

[49] A. M. Khan, H. El-Daly, and N. M. Rajpoot. A gamma-gaussian mixture model for detection of mitotic cells in breast cancer histopathology images. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 149–152, Nov 2012.

[50] D. Kong, D. Gray, and Hai Tao. A viewpoint invariant approach for crowd counting. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 1187–1190, 2006.

[51] Hui Kong, Hatice Cinar Akakin, and Sanjay E. Sarma. A generalized laplacian of gaussian filter for blob detection and its applications. *IEEE Transactions on Cybernetics*, 43:1719–1733, 2013.

[52] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classication with deep convolutional neural networks. *Neural Information Processing Systems*, 2012.

[53] Logan Lebanoff and Haroon Idrees. Counting in dense crowds using deep learning. *CRCV*, 2015.

[54] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–444, 2015.

[55] Honglak Lee, Chaitanya Ekanadham, and Andrew Y. Ng. Sparse deep belief net model for visual area v2. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 873–880. Curran Associates, Inc., 2008.

[56] Bastian Leibe, Aleš Leonardis, and Bernt Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1):259–289, May 2008.

[57] Victor Lempitsky and Andrew Zisserman. Learning to count objects in images. *Neural Information Processing Systems.*, 2010.

[58] Zhe Lin and Larry S. Davis. Shape-based human detection and segmentation via hierarchical part-template matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, page 604618, 2010.

[59] Fujun Liu and Lin Yang. A novel cell detection method using deep convolutional neural network and maximum-weight independent set. *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[60] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[61] Chongyang Ma, Li-Yi Wei, and Xin Tong. Discrete element textures. *ACM Trans. Graph.*, 30(4):62:1–62:10, July 2011.

[62] Christopher D. Malon and Eric Cosatto. Classification of mitotic figures with convolutional neural networks and seeded blob features. 2013.

[63] Christopher D. Malon and Eric Cosatto. Classification of mitotic figures with convolutional neural networks and seeded blob features. *Journal of Pathology Informatics*, 2013.

[64] A. N. Marana, S. A. Velastin, L. F. Costa, and R. A. Lotufo. Estimation of crowd density using image processing. In *IEE Colloquium on Image Processing for Security Applications (Digest No: 1997/074)*, pages 11/1–11/8, Mar 1997.

[65] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *Proceedings of the British Machine Vision Conference*, pages 36.1–36.10. BMVA Press, 2002. doi:10.5244/C.16.36.

[66] Oquab Maxime, Bottou Leon, Laptev Ivan, and Sivic Josef. Learning and transferring mid-level image representations using convolutional neural networks. 2014.

[67] Adam Krzyak Mehdi Habibzadeh and Thomas Fevens. White blood cell differential counts using convolutional neural networks for low resolution images. *Artificial Intelligence and Soft Computing*, pages 263–274, 2013.

[68] Erik Meijering. Neuron tracing in perspective. 77:693–704, 07 2010.

[69] Tim Nattkemper, Heiko Wersing, Walter Schubert, and Helge Ritter. A neural network architecture for automatic segmentation of fluorescence micrographs. 48:357–367, 10 2002.

[70] T. Ojala, M. Pietikinen, and D. Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29:51–59, 1996.

[71] Judith M. S. Prewitt and Mortimer L. Mendelsohn. The analysis of cell images. *Annals of the New York Academy of Sciences*, 128:1035–1053, 1966.

[72] Vincent Rabaud and Serge Belongie. Counting crowded moving objects. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, CVPR '06, pages 705–711, Washington, DC, USA, 2006. IEEE Computer Society.

[73] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[74] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 2015.

[75] Jens Rittscher. Characterization of biological processes through automated image analysis. *Annual Review of Biomedical Engineering*, 12(1):315–344, 2010. PMID: 20482277.

[76] Camp RL, Chung GG, and Rimm DL. Automated subcellular localization and quantification of protein expression in tissue microarrays. *Nature Medicine.*, pages 1323–7, 2002.

[77] Ludovic Roux, Daniel Racoceanu, N. Loménie, Maria S. Kulikova, Humayun Irshad, J. Klossa, Frédérique Capron, Catherine Genestie, Gilles Le Naour, and Metin N. Gurcan. Mitosis detection in breast cancer histological images, an icpr 2012 contest. *Journal of Pathology Informatics*, 4, 05/2013 2013.

[78] David Ryan, Simon Denman, Clinton Fookes, and Sridha Sridharan. Crowd counting using multiple local features. In *Proceedings of the 2009 Digital Image Computing: Techniques and Applications*, DICTA '09, pages 81–88, Washington, DC, USA, 2009. IEEE Computer Society.

[79] Oriol Pujol Santi Segu and Jordi Vitri. Learning to count with deep object features. *Computer Vision and Pattern Recognition.*, 2015.

[80] Albarqouni Shadi, Baur Christoph, Achilles Felix, Belagiannis Vasileios, Demirci Stefanie, and Navab Nassir. Aggnet: Deep learning from crowds for mitosis detection in breast cancer histology images. *IEEE Transactions on Medical Imaging.*, 2016.

[81] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional models for semantic segmentation. *The IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[82] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[83] Christoph Sommer, Luca Fiaschi, F.A. Hamprecht, and Daniel Gerlich. Learning-based mitotic cell detection in histopathological images. pages 2306–2309, 01 2012.

[84] Hai Su, Fuyong Xing, Xiangfei Kong, Yuanpu Xie, Shaoting Zhang, and Lin Yang. Robust cell detection and segmentation in histopathological images using sparse reconstruction and stacked denoising autoencoders. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 383–390, Cham, 2015. Springer International Publishing.

[85] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[86] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[87] F. Boray Tek. Mitosis detection using generic features and an ensemble of cascade adaboosts. In *Journal of pathology informatics*, 2013.

[88] Jr; Jeffrey D. Messinger; Tianjiao Zhang; Mark J. Bentley; Danielle B. Gutierrez; Zsolt Ablonczy; R. Theodore Smith; Kenneth R. Sloan; Christine A. Curcio Thomas

Ach; Carrie Huisingh; Gerald McGwin. Quantitative autofluorescence and cell density maps of the human retinal pigment epithelium. *Investigative Ophthalmology Visual Science*, 2014.

[89] W. E. Tolles. The cytoanalyzer-an example of physics in medical research. *Transactions of the New York Academy of Sciences*, 17 3:250–6, 1955.

[90] Ryota Tomioka, Taiji Suzuki, and Masashi Sugiyama. Super-linear convergence of dual augmented lagrangian algorithm for sparsity regularized estimation. *The Journal of Machine Learning Research*, 2011.

[91] Karunya Tota and Haroon Idrees. Counting in dense crowds using deep features. *CRCV*, 2015.

[92] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Random k-labelsets for multi-label classification. *IEEE Transactions on Knowledge and Data Engineering*, pages 1079–1089, 2011.

[93] Mitko Veta, P J. van Diest, and J P. W. Pluim. Detecting mitotic figures in breast cancer histopathology images. 8676:867607, 03 2013.

[94] Mitko Veta, Paul J van Diest, Stefan M Willems, Haibo Wang, Anant Madabhushi, Angel Cruz-Roa, Fabio Gonzalez, Anders B L Larsen, Jacob S Vestergaard, Anders B Dahl, Dan C Cireşan, Jürgen Schmidhuber, Alessandro Giusti, Luca M Gambardella, F Boray Tek, Thomas Walter, Ching-Wei Wang, Satoshi Kondo, Bogdan J Matuszewski, Frederic Precioso, Violet Snell, Josef Kittler, Teofilo E de Campos, Adnan M Khan, Nasir M Rajpoot, Evdokia Arkoumani, Miangela M Lacle, Max A Viergever, and Josien P W Pluim. Assessment of algorithms for mitosis detection in breast cancer histopathology images. *Medical image analysis*, 20:237–48, 02/2015 2015.

[95] C. Vonesch, F. Aguet, J. L. Vonesch, and M. Unser. The colored revolution of bioimaging. *IEEE Signal Processing Magazine*, 23(3):20–31, May 2006.

[96] Haibo Wang, Angel Cruz-Roa, Ajay Basavanhally, Hannah Gilmore, Natalie Shih, Mike Feldman, John Tomaszewski, Fabio Gonzalez, and Anant Madabhushi. Cascaded ensemble of convolutional neural networks and handcrafted features for mitosis detection. In *SPIE Medical Imaging*, volume 9041, pages 90410B–90410B–10, 03/2014 2014.

[97] Meng Wang and Xiaogang Wang. Automatic adaptation of a generic pedestrian detector to a specific traffic scene. In *CVPR*, pages 3401–3408. IEEE Computer Society, 2011.

[98] Meng Wang and Xiaogang Wang. Automatic adaptation of a generic pedestrian detector to a specific traffic scene. In *CVPR*, pages 3401–3408. IEEE Computer Society, 2011.

[99] Bo Wu and Ram Nevatia. Detection and tracking of multiple, partially occluded humans by bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75(2):247–266, Nov 2007.

[100] Bo Wu and Ram Nevatia. Detection and segmentation of multiple, partially occluded objects by grouping, merging, assigning part detection responses. *International Journal of Computer Vision*, 82(2):185–204, Apr 2009.

[101] Weidi Xie, J. Alison Noble, and Andrew Zisserman. Microscopy cell counting with fully convolutional regression networks. *Deep Learning Workshop in MICCAI*, 2015.

[102] Yao Xue, Nilanjan Ray, Judith Hugh, and Gilbert Bigras. Cell counting by regression using convolutional neural network. *ECCV 2016 workshop on BioImage Computing*, 2016.

[103] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *in IEEE Conference on Computer Vision and Pattern Recognition(CVPR*, 2009.

[104] Xiaonong Zhang Yifan Chen. Study of cell behaviors on anodized tio 2 nanotube arrays with coexisting multi-size diameters. *Nano-Micro Letters*, 2015.

[105] Kai Yu, Tong Zhang, and Yihong Gong. Nonlinear learning using local coordinate coding. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 2223–2231. Curran Associates, Inc., 2009.

[106] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *The European Conference on Computer Vision (ECCV)*, pages 818–833, 2014.

[107] Tao Zhao and R. Nevatia. Bayesian human segmentation in crowded situations. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–459–66 vol.2, June 2003.

Appendix

Before starting the mathematical derivation of gradients, let us first look at a related optimization problem, ridge regression, which explains why we needed to derive backpropagation rules carefully for the L_1 recovery problem. The ridge regression problem solves the following optimization:

$$\arg\min_a \parallel x - Da \parallel_2^2 + \lambda \parallel a \parallel_2^2 \tag{6.1}$$

For the ridge regression, there is a closed form solution:

$$a = \left(D^T D + \lambda I\right)^{-1} D^T x \tag{6.2}$$

Thinking from the perspective of training a neural network, the above solution directly tells us about how a is computed through the forward pass. Furthermore, we can derive a back propagation rule based on the closed form solution. The output code a can be viewed as a function of input code x. So we have the Jacobian:

$$J_x^a = D\left(D^T D + \lambda I\right)^{-1} \tag{6.3}$$

and the partial derivative of the cost with respect to x:

$$\delta x = J_x^a \delta a = D\left(D^T D + \lambda I\right)^{-1} \delta a. \tag{6.4}$$

For the L_1 minimization problem there is no closed form solution for the forward pass. The central idea of L_1 minimization algorithm is to find an optimized vector a according to the following loss function:

$$\arg\min_a \frac{1}{2} \parallel x - Da \parallel_2^2 + \lambda \parallel a \parallel_1, \tag{6.5}$$

where $D \in \mathbb{R}^{m \times n}$, $a \in \mathbb{R}^{n \times 1}$, $x \in \mathbb{R}^{m \times 1}$.

To derive backpropagation for (6.5), we replace the L_1 norm by a smooth, convex approximation $f(a)$. An example smooth approximation is $f(a) = \sqrt{a^2 + \epsilon^2}$, where ϵ is a small positive number. So, our minimization problem becomes:

$$\arg\min_a \frac{1}{2} \parallel x - Da \parallel_2^2 + \lambda f(a) \tag{6.6}$$

To get the a that minimizes the 6, the necessary and sufficient condition is given as:

$$D^T (Da - x) + \lambda f^{'}(a) = 0 \tag{6.7}$$

Derivation of partial derivative δx

Now differentiating the equation (6.7) with respect to x, we get

$$D^T D \left(\frac{\partial a}{\partial x}\right) + \lambda diag\left(f^{''}\right)\left(\frac{\partial a}{\partial x}\right) = D^T \tag{6.8}$$

In other words,

$$\left(\frac{\partial a}{\partial x}\right)^T = D\left[D^T D + \lambda diag\left(f^{''}\right)\right]^{-1} \tag{6.9}$$

so

$$\delta x = \left(\frac{\partial a}{\partial x}\right)^T \delta a \tag{6.10}$$

$\left(\frac{\partial a}{\partial x}\right)^T$ is the Jacobian matrix, so that

$$\frac{\partial a}{\partial x} = \begin{bmatrix} \frac{\partial a_1}{\partial x_1} & \cdots & \frac{\partial a_n}{\partial x_1} \\ \frac{\partial a_1}{\partial x_2} & \cdots & \frac{\partial a_n}{\partial x_2} \\ \vdots & \ddots & \vdots \\ \frac{\partial a_1}{\partial x_n} & \cdots & \frac{\partial a_n}{\partial x_n} \end{bmatrix} \tag{6.11}$$

$f^{'}(a)$ is defined as $f^{'}(a) = \begin{bmatrix} \frac{\partial f}{\partial a_1} & \frac{\partial f}{\partial a_2} & \cdots & \frac{\partial f}{\partial a_n} \end{bmatrix}^T$ and $f^{''}(a)$ is a diagonal matrix:

$$f^{''}(a) = \begin{bmatrix} \frac{\partial^2 f}{\partial a_1^2} & 0 & \cdots & 0 \\ 0 & \frac{\partial^2 f}{\partial a_2^2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\partial^2 f}{\partial a_n^2} \end{bmatrix} \tag{6.12}$$

Then we have

$$\frac{\partial a}{\partial x} = D\left[D^T D + \lambda f^{''}(a)\right]^{-1} \tag{6.13}$$

Let p be any set of indices. Let us denote by $A(:,p)$ all the column of matrix A, whose indices belong to the set p. Similarly, $A(q,p)$ denotes the sub-matrix of the sequence matrix A, where rows and column indices belong to the sets q and p, respectively.

Now let p and q denote two specific sets: $p = \{i : a_i \neq 0\}$ and $q = \{i : a_i = 0\}$. So now we can rewrite (6.13) as:

$$\begin{bmatrix} \frac{\partial a}{\partial x}(:,p) & \frac{\partial a}{\partial x}(:,q) \end{bmatrix} = \begin{bmatrix} D(:,p) & D(:,q) \end{bmatrix} \begin{bmatrix} D^T D(p,p) + \lambda f^{''}(a)(p,p) & D^T D(p,q) \\ D^T D(q,p) & D^T D(q,q) + \lambda f^{''}(a)(q,q) \end{bmatrix}^{-1} \tag{6.14}$$

Using Schur's complement, we can get that

$$\begin{bmatrix} D^TD\left(p,p\right)+\lambda f^{''}\left(a\right)\left(p,p\right) & D^TD\left(p,q\right) \\ D^TD\left(q,p\right) & D^TD\left(q,q\right)+\lambda f^{''}\left(a\right)\left(q,q\right) \end{bmatrix}^{-1} =$$
$$\begin{bmatrix} I_p & 0 \\ -\left[D^TD\left(q,q\right)+\lambda f^{''}\left(a\right)\left(q,q\right)\right]^{-1} D^TD\left(q,p\right) & I_q \end{bmatrix}$$
$$\begin{bmatrix} U & 0 \\ 0 & \left[D^TD\left(q,q\right)+\lambda f^{''}\left(a\right)\left(q,q\right)\right]^{-1} \end{bmatrix} \begin{bmatrix} I_p & -D^TD(p,q)\left[D^TD\left(q,q\right)+\lambda f^{''}\left(a\right)\left(q,q\right)\right]^{-1} \\ 0 & I_q \end{bmatrix} \tag{6.15}$$

where $U = \left(D^TD\left(p,p\right)+\lambda f^{''}\left(a\right)\left(p,p\right) - D^TD\left(p,q\right)\left[D^TD\left(q,q\right)+\lambda f^{''}\left(a\right)\left(q,q\right)\right]^{-1} D^TD\left(q,p\right)\right)^{-1}$; I_p and I_q are identity matrices of order $|p|$ and $|q|$, respectively.

Note that $\lambda f^{''}\left(a\right)\left(p,p\right) \to 0$ and $\lambda f^{''}\left(a\right)\left(q,q\right)$ becomes diagonal matrix with diagonal entries trending to infinity. This property is actually a common property of any reasonable L_1 norm approximation functions. For example, $f_\epsilon\left(a\right) = \sqrt{a^2+\epsilon}$ is a widely-used approximation function, whose second order derivative is $f_\epsilon^{''}\left(a\right) = \frac{1}{\sqrt{a^2+\epsilon}} - \frac{a^2}{\left(a^2+\epsilon\right)^{3/2}}$. It is easy to find that its second order derivative satisfies the property that: $f_\epsilon^{''}\left(a\right) \Longrightarrow \infty$, when $\epsilon \to 0, a = 0$; and $f_\epsilon^{''}\left(a\right) \Longrightarrow 0$ when $\epsilon \to 0, a \neq 0$. Thus, we also have

$$\left[D^TD\left(q,q\right)+\lambda f^{''}\left(a\right)\left(q,q\right)\right]^{-1} \longrightarrow 0 \tag{6.16}$$

Combine these results together, we get:

$$\begin{bmatrix} D^TD\left(p,p\right)+\lambda f^{''}\left(a\right)\left(p,p\right) & D^TD\left(p,q\right) \\ D^TD\left(q,p\right) & D^TD\left(q,q\right)+\lambda f^{''}\left(a\right)\left(q,q\right) \end{bmatrix}^{-1} \longrightarrow \begin{bmatrix} \left[D^TD\left(p,p\right)\right]^{-1} & 0 \\ 0 & 0 \end{bmatrix} \tag{6.17}$$

So, we have

$$\begin{aligned} \begin{bmatrix} \frac{\partial a}{\partial x}\left(:,p\right) & \frac{\partial a}{\partial x}\left(:,q\right) \end{bmatrix} &= \begin{bmatrix} D\left(:,p\right) & D\left(:,q\right) \end{bmatrix} \begin{bmatrix} \left[D^TD\left(p,p\right)\right]^{-1} & 0 \\ 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} D\left(:,p\right)\left[D^TD\left(p,p\right)\right]^{-1} & 0 \end{bmatrix} \end{aligned} \tag{6.18}$$

Now suppose $\delta a \equiv$ gradient of loss with respect to a, then gradient of loss with respect to x is

$$\delta x = \begin{bmatrix} \frac{\partial a}{\partial x}\left(:,p\right) & \frac{\partial a}{\partial x}\left(:,q\right) \end{bmatrix} \begin{bmatrix} \delta a\left(p\right)^T & \delta a\left(q\right)^T \end{bmatrix} \tag{6.19}$$

$$\delta x = D\left(:,p\right)\left[D^TD\left(p,p\right)\right]^{-1}\delta a\left(p\right) \tag{6.20}$$

Derivation of partial derivative δD

Let's start from the equation:

$$D^TDa + \lambda f^{'}\left(a\right) = D^Tx \tag{6.21}$$

Let us partition $D_{m\times n}$ as $D_{m\times n} = \begin{bmatrix} D_1 & D_2 & \dots & D_n \end{bmatrix}$ where $D_i's$ are $m\times 1$ column vectors. With these notations, the above equation can be written as:

$$D_i^T \sum_{j=1}^{n} D_j a_j + \lambda f^{'}\left(a_i\right) = D_i^T x \ \ \forall i = 1, 2, \dots, n \tag{6.22}$$

Differentiating (6.22) with respect to D_i, we obtain:

$$\sum_{j=1}^{n} D_j a_j + D_i a_i + \sum_{j=1}^{n} D_i^T D_j \frac{\partial a_j}{\partial D_i} + \lambda f^{''}\left(a_i\right) \frac{\partial a_i}{\partial D_i} = x \ \ \forall i = 1, 2, \dots, n \tag{6.23}$$

Differentiating (6.22) with respect to D_k, we obtain:

$$D_i a_k + \sum_{j=1}^{n} D_i^T D_j \frac{\partial a_j}{\partial D_k} + \lambda f^{''}\left(a_i\right) \frac{\partial a_i}{\partial D_k} = 0, \begin{cases} i = 1, 2, \dots, n, \\ k = 1, 2, \dots, n, \\ i \neq k. \end{cases} \tag{6.24}$$

Interchanging indices i and k, we obtain:

$$D_k a_i + \sum_{j=1}^{n} D_k^T D_j \frac{\partial a_j}{\partial D_i} + \lambda f^{''}\left(a_k\right) \frac{\partial a_k}{\partial D_i} = 0, \begin{cases} i = 1, 2, \dots, n, \\ k = 1, 2, \dots, n, \\ i \neq k. \end{cases} \tag{6.25}$$

Combining (6.23) and (6.25), we obtain:

$$\frac{\partial a_i}{\partial D_i} \left[D^T D + \lambda diag\left(f^{''}\right)\right] = \begin{bmatrix} -D_1 a_i & -D_2 a_i & \dots & x - Da - D_i a_i & \dots & -D_n a_i \end{bmatrix} \tag{6.26}$$

Here $\dfrac{\partial a_i}{\partial D_i}$ is defined as:

$$\left(\frac{\partial a}{\partial D_i}\right)_{m\times n} = \begin{bmatrix} \dfrac{\partial a_1}{\partial D_i} & \dfrac{\partial a_2}{\partial D_i} & \dots & \dfrac{\partial a_n}{\partial D_i} \end{bmatrix}_{m\times n} \tag{6.27}$$

So,

$$\frac{\partial a_i}{\partial D_i} = \begin{bmatrix} -D_1 a_i & -D_2 a_i & \dots & x - Da - D_i a_i & \dots & -D_n a_i \end{bmatrix} \left[D^T D + \lambda diag\left(f^{''}\right)\right]^{-1} \tag{6.28}$$

Now using the chain rule, we get:

$$\begin{aligned} \delta D_i \equiv \bigtriangledown_{Di}\left(loss\right) &= \left(\frac{\partial a_i}{\partial D_i}\right)_{m\times n} \delta a_{n\times 1} \\ &= \begin{bmatrix} -D_1 a_i & -D_2 a_i & \dots & x - Da - D_i a_i & \dots & -D_n a_i \end{bmatrix} \left[D^T D + \lambda diag\left(f^{''}\right)\right]^{-1} \delta a \end{aligned} \tag{6.29}$$

where $i = 1, 2, \dots, n$ and $\delta a \equiv \bigtriangledown_a \left(loss\right)$.

Collecting these n equations in a matrix

$$\delta D = \left(x - Da\right) \delta a^T \left[D^T D + \lambda diag\left(f^{''}\right)\right]^{-1} - D \left[D^T D + \lambda diag\left(f^{''}\right)\right]^{-1} \delta a a^T \tag{6.30}$$

using previous notations, we arrive at:

$$\delta D = \left(x - Da\right) \delta a\left(p\right)^T \left[D^T D\left(p, p\right)\right]^{-1} - D(:, p) \left[D^T D\left(p, p\right)\right]^{-1} \delta a\left(p\right) a\left(p\right)^T \tag{6.31}$$