

A DEEP LEARNING ARCHITECTURE FOR HISTOLOGY IMAGE  
CLASSIFICATION

Chia-Yu Kao

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill  
2018

Approved by:

Leonard McMillan

Vladimir Jovic

Marc Niethammer

Deborah O'Brien

Fernando Pardo Manuel de Villena

©2018  
Chia-Yu Kao  
ALL RIGHTS RESERVED

## **ABSTRACT**

Chia-Yu Kao: A Deep Learning Architecture for Histology Image Classification  
(Under the direction of Leonard McMillan)

Over the past decade, a machine learning technique called deep-learning has gained prominence in computer vision because of its ability to extract semantics from natural images. However, in contrast to the natural images, deep learning methods have been less effective for analyzing medical histology images. Analyzing histology images involves the classification of tissue according to cell types and states, where the differences in texture and structure are often subtle between states. These qualitative differences between histology and natural images make transfer learning difficult and limit the use of deep learning methods for histology image analysis.

This dissertation introduces two novel deep learning architectures, that address these limitations. Both provide intermediate hints to aid deep learning models. The first deep learning architecture is constructed based on stacked autoencoders with an additional layer, called a hyperlayer. The hyperlayer is an intermediate hint that captures image features at different scales. The second architecture is a two-tiered Convolutional Neural Networks (CNN), with an intermediate representation, called a pixel/region labeling. The pixel/region labels provide a normalized semantic description that can be used as an input to a subsequent image classifier.

The experiments show that by adding the hyperlayer, the architecture substantially outperforms fine-tuned CNN models trained without an intermediate target. In addition, the experiments suggest that the advantages of the labeling classifier are threefold. First, it generalizes to other related vision tasks. Second, image classification does not require extremely accurate pixel labeling. The architecture is robust and not susceptible to the noise. Lastly, labeling model captures low-level texture information and converts them to valuable hints.

## ACKNOWLEDGEMENTS

First and foremost, I want to give my sincere thank to my advisor Leonard McMillan. I am very grateful for the time, advice, knowledge, and life experiences he has generously shared with me, especially grateful for his patience in guiding me and helping me overcome the language gap and culture shock in my first few years.

I would also like to thank my committee members for graciously agreeing to serve on my committee. I have been fortunate to collaborate with Deborah O'Brien and Fernando Pardo Manuel de Villena, whose deep insights into biological topics amaze me. I especially appreciate Debbie for spending time on teaching me how to interpret those histology images and recognize the cell types. My sincere thank to Vladimir Jovic and Marc Niethammer for the support and the insightful discussions that help enrich this research. I also want to give my special thank to Timothy Bell for collecting all these mouse testis, David Aylor and John Shorter who have been interested in my research, Darla Miller who supplies all the mice and graciously provides all the mice and other tissues, and O'Brien's lab for providing the histology classification dataset.

I have had a fantastic group of lab mates throughout the years – Jeremy Wang, Catie Welsh, Shunping Huang, Ping Fu, Matt Holt, Zhaoxi Zhang, Piyush Aggarwal, Seth Greenstein, Maya Najarian, Anwica Kashfeen, Sebastian Sigmon, Bhavya Vyas, and Mia Madduri. Thank you all for making the lab such an enjoyable place to work and making these five and half years as a happy and joyful journey.

Finally, I want to give thanks to my parents. I want to thank them for giving me a wonderful childhood and encouraging me to follow my dream. There is no me without you. I would like to thank my sweet boy Nate for being such a wonderful boy always cheering me up. And most of all for my loving, supportive, encouraging, and patient husband Derek whose faithful support during the final stages of this journey is so appreciated. Love you, Derek.

## TABLE OF CONTENTS

LIST OF TABLES .....	ix
LIST OF FIGURES .....	xi
1 Introduction .....	1
1.1 Motivation .....	2
1.2 Thesis Statement .....	4
1.3 Outline of Contributions .....	5
2 Background and Related Work .....	7
2.1 Introduction .....	7
2.2 Classical Neural Networks .....	7
2.3 Deep Learning .....	8
2.3.1 Convolutional Neural Networks (CNN) .....	9
2.3.2 Autoencoder .....	11
2.3.3 Curriculum Learning .....	12
2.4 Automated Histology Image Systems .....	13
2.5 Deep Learning on Automated Histology Image Systems .....	13
3 Using a Hyperlayer to Combine Features Across Scales .....	15
3.1 Introduction .....	15
3.2 Approach .....	17
3.2.1 Feature Learning via Stacked Convolutional Autoencoder .....	17
3.2.2 Hyperlayer construction .....	19
3.2.3 Automated Classification via Softmax Classifier .....	19

3.2.4	Image Representation via Deconvolutional Networks .....	19
3.3	Experimental Setup.....	20
3.3.1	Histology Image Dataset .....	20
3.3.2	Building the Classifier .....	21
3.3.3	Classification Performance and Comparison.....	22
3.4	Experimental Results .....	22
3.4.1	Automatic Classification Performance .....	23
3.4.2	Image Reconstruction with Deconvolutional Networks.....	24
3.5	Conclusion.....	25
4	Pixel Labeling as an Intermediate Learning Objective.....	27
4.1	Introduction.....	27
4.2	Approach .....	28
4.2.1	Pixel Labeling Layer .....	29
4.2.1.1	Supervised Pixel Labels .....	30
4.2.1.2	Window Selection .....	30
4.2.1.3	Data Normalization .....	31
4.2.1.4	Pixel Labeling Model.....	31
4.2.2	Image Reference.....	33
4.2.3	Image Classification .....	33
4.3	Experiment Setup .....	34
4.3.1	Histology Image Dataset .....	34
4.3.2	Pixel Labeling Layer .....	35
4.3.3	Image Classification Layer .....	36
4.4	Experiment Results.....	40
4.4.1	Pixel Labeling Results .....	41
4.4.2	Sensitivity Analysis of Pixel Labeling Classifier.....	42

4.4.3	Classification Results .....	44
4.5	Conclusion and Discussion .....	45
5	Generalizing Pixel Labeling Approach .....	47
5.1	Introduction .....	47
5.2	Pentomino Dataset .....	48
5.2.1	Problem Definition .....	48
5.2.2	Experiment Setup .....	49
5.2.2.1	Block Size Selection .....	49
5.2.2.2	Original Model Architecture .....	50
5.2.2.3	Pixel Labeling Layer .....	53
5.2.3	Experiment Results .....	55
5.2.3.1	Pixel Labeling Hints .....	55
5.2.3.2	Sensitivity Analysis on Pixel Labeling Classifier .....	59
5.3	Brodatz Texture Dataset .....	59
5.3.1	Problem Definition .....	60
5.3.2	Experiment Setup .....	62
5.3.2.1	New Texture Image Generation .....	62
5.3.2.2	Region Labeling Hints .....	64
5.3.3	Experiment Results .....	65
5.3.4	Optimize Two Problems with Joint Objective Functions .....	67
5.4	Conclusion .....	69
6	Unsupervised Pixel Labeling Layer .....	71
6.1	Introduction .....	71
6.2	Approach .....	72
6.2.1	Probability Distribution .....	72
6.2.2	Correlation Relationships .....	75

6.2.3	Loss Function Definition .....	76
6.2.3.1	Uncertainty .....	77
6.2.3.2	Sparsity.....	79
6.2.3.3	Relationship Between Uncertainty and Sparsity .....	83
6.2.3.4	Distinction .....	86
6.2.3.5	Relationships of Distinction and Sparsity to Other Correlation Measures .....	90
6.2.3.6	Final Loss Function .....	92
6.3	Experimental Results .....	94
6.3.1	Evaluation on Regularization Terms .....	94
6.3.2	Comparison Between Sparsity and Distinction.....	97
6.4	Conclusion and Discussion .....	99
7	Conclusions and Future Work .....	100
7.1	Extensions to Combine Features Across Scales .....	100
7.2	Extension to Supervised Pixel/Region Labeling .....	101
7.3	Conclusion.....	105
	BIBLIOGRAPHY.....	106

## LIST OF TABLES

3.1	The hyperparameter settings for all layers in the learning architecture. Layer type: I - input, C - convolutional layer, MP - max-pooling layer, H - hyper layer, FC - fully-connected hidden layer, LG - logistic regression layer. ....	22
3.2	Comparison of classification performance on the first dataset (normal / many vacuoles), where SCAE is Stacked Convolutional AutoEncoders. I compared the method with traditional CNN and SCAE with same parameters. The best results are in bold typeface. Accuracy is the ratio of correctly predicted observation to the total observations. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. Recall measures the proportion of positives that are correctly identified, while specificity measures the proportion of negatives that are correctly identified. ....	23
3.3	Comparison of classification performance on the second dataset (normal / many vacuoles / last stage of germ cell loss / many vacuoles and last stage of germ cell loss). The hyperparameters are same in these three methods. The best results are in bold typeface. ....	23
3.4	Examples of our models classification results compared to the ground truth. The normal tubules that the model identified with vacuoles are arguable since most appear to exhibit vacuole-like features. ....	24
3.5	Examples of reconstructed images generated from the deconvolutional networks by taking the hidden representation of the particular layer and cascading it back to the original image. Notice how the reconstructed image becomes blurrier at higher layer because max-pooling and convolution tend to decrease the resolution and smooth the images at higher layers. ....	25
4.1	The configuration details of our two-tiered architecture, pixel labeling model and image classification model. Layer type: Conv - convolutional layer, FC - fully-connected layer. ....	32
4.2	Tubule count distribution over all categories. ....	38
4.3	Pixel labeling results. In the top 3 scores, our pixel labeling model can predict above 90% accuracy. ....	40
4.4	Sensitivity analysis of pixel labeling classifier on five categories, where rows are the corrupted rate on pixel labeling results. ....	42
4.5	Comparison of classification performance on six datasets. I compared my method with VGG-16 and CNN without hints. The best results are in bold typeface.....	44

5.1	Image classification results of two SMLP learning processes in 20k dataset compared with applying pixel hints or not. ....	57
5.2	Sensitivity analysis of pixel labeling hints and P1NN hints on image classification task. Columns represent the corruption rates applied to the pixel labeling results. ....	58
5.3	Classification results on generated texture dataset with different levels hints provision. .	66

## LIST OF FIGURES

1.1	(a) Natural scene images from Pascal VOC 2012 (Everingham et al., 2011). (b) Stained histology images of normal/abnormal testis tubules. The objects in natural images generally appear at different depths, and are distinct from the background. In histology images, the tissue is at a single depth. Compared to the natural scene images, texture detailed information is essential to define an object. . . .	2
2.1	A sample unit $i$ with input dimension $d$ . . . . .	8
3.1	A composite testis image from lab mice with normal seminiferous tubules. Tubules 1, 2, 3, and 4 are shown at greater magnification. . . . .	16
3.2	Examples of tubule with different abnormalities. (a) a normal tubule (b) a tubule with many vacuoles (c) a tubule with abnormal germ cells (red arrow) (d) tubule with complete germ cell loss and one large vacuole (few vacuoles in the manual labeling scheme) (e) a tubule with germ cells in lumen. . . . .	16
3.3	(a) Overview of our deep learning architecture, where the building blocks within green box are convolutional auto-encodes performed in the first stage of classification, and all building blocks within blue box compose a whole learning framework for the fine-tuning stage. Deconvolutional networks indicated in green dotted lines goes through whole framework to reconstruct the image from the top most layer. (b) Example of a hyper layer with weights 3, 2, and 1 on convolutional layer 1, 2, and 3, respectively. (c) Example of up-sampling operation on pool size $2 \times 2$ in deconvolutional networks. Pool-sized neighborhoods are filled with the absolute max value within that area. . . . .	18
4.1	(a)(c) Examples of true-color testis histology cross sections of normal and abnormal tubules. (b)(d) False-color images indicating possible cell types at each pixel. . . . .	28
4.2	The classification task is divided into two subtasks, a per-pixel labeling and an image classification. The pixel labeling model is trained using a CNN trained on $23 \times 23$ image patches, whose output is a likelihood function of 18 possible labels. Once learned, this model is applied to every pixel of an image ( $65 \times 65$ ) and then used as input to train the image classifier. . . . .	29
4.3	Supervised labels are predefined for characterizing each pixel. The labels define the structure of a testis from outside of the testis to the center of the tubule, varied from different kinds of empty space to specific cell types. . . . .	30

4.4	Above image is the screenshot of interactive tool designed for labeling patches. The squares with different colors represent 18 predefined labels. Images below are the patch examples in the labeling dataset, where black cross indicates the center of the tubule. ....	35
4.5	The tubule-classification dataset includes five tubule categories: normal, vacuoles, abnormal germ cell, germ cell loss, as well as germ cell in lumen. Different text background colors represent varied degree of individual category. The dataset of each experiment contains different combinations of five categories. Five experiments include (1) normal / many vacuoles (2) normal / few / many vacuoles (3) normal / Sertoli cell only (4) five stages of germ cell loss (5) normal / germ cell in lumen. The table lists the combinations of five categories for each class defined in five experiments. ....	37
4.6	CNN without hints: the classification task is trained without intermediate target and directly backpropagate the parameters from top to bottom. ....	39
4.7	Confusion matrix of pixel labeling. Rows are the ground truth, and the columns are the predicted results. ....	41
4.8	Visualizations of the learned pixel labeling model. (a) Example of a true-color testis histology cross section. (b)(c) The probability map for a single pixel-label class (basal lamina and vacuole). Brighter colors indicate higher probabilities. (d) is a false-color image indicating the label with the highest probability at each pixel where pink represents vacuoles, yellow indicates lumen and magenta is tunica. Pink, yellow and magenta clearly highlight regions where the pixel labeling is accurate. Concentric rings of labels within tubules are also mostly as expected. ....	43
4.9	Examples of our classification results compared to the ground truth. The examples in first two columns are normal tubules, followed by tubules with few vacuoles, and tubules with many vacuoles. The normal tubules that the model identified with vacuoles are pardonable since most appear to exhibit vacuole-like features. The tubules with vacuoles that the model classified as normal tended to have few vacuoles, thus, leading to ambiguity. ....	45

4.10	Image classification results of five experiments. The results of five experiments are arranged in five rows. The first five columns show the examples of true positive and true negative, while the false positive/negative examples are illustrated in the last five columns. The text below image shows the class of the truth versus prediction, where blue means correct prediction while red is wrong. The numbers are different classes within each experiment defined in Figure 4.5. In the second and forth experiments, they include degrees of abnormalities, leading to the difficulty to distinguish the boundary cases among states. For instance, image in row four and column seven has features similar to the confused class 0, and image in column eight and class 1 are very much alike. ....	46
5.1	(a) Different types of Pentominoes used in the dataset (Gülçehre and Bengio, 2016). Example images from our modified dataset showing (b) same Pentomino type and (c) different Pentomino types.....	49
5.2	Red square in the images are example of aperture windows using in the pixel labeling model. The pixel labeling model predicts the possible Pentomino type on the center pixel of the aperture window in the context of its surrounding neighbors...	50
5.3	(a) Structured MLP architecture with hints (SMLP-Hints) (Gülçehre and Bengio, 2016). P1NN are bottom two layers and top two layers are P2NN. P1NN and P2NN are trained separately, not jointly. In SMLP-Hints, P1NN is trained on each $8 \times 8$ patch extracted from the image and the softmax output probabilities of all 64 patches are concatenated into a $64 \times 11$ vector that forms the input of P2NN. (b) Structured MLP architecture without hints (SMLP-Nohints) (Gülçehre and Bengio, 2016). It is the same architecture as SMLP-Hints but with P1NN and P2NN trained jointly in terms of the final classification task. ....	51
5.4	SMLP architecture using pixel labeling hints. Bottom two layers are pixel labeling layers, while middle two and top two layers are P1NN and P2NN, respectively. Pixel labeling layer is trained on $5 \times 5$ patch extracted from the image and outputs probabilities for each of the 11 possible Pentomino at every pixel in the image. ....	52
5.5	(a) Examples of background patch pattern. (b) An example of patch pattern that is unique to Pentomino 2. (c) An example of patch pattern that is not unique to any one Pentomino. Both Pentomino 4 and 9 could have such pattern, leading to ambiguity while labeling the pixels. ....	54

5.6	Confusion matrix of the pixel labeling on all transformations of Pentominoes. Columns are the predicted results of the pixel labeling model, and rows are the ground truth of ten Pentominoes with eight transformations plus background (10), where the numbers represent ten Pentomino type (0-9) and the letters indicate the eight transformations (a-d, A-D). Lowercase (a-d) are small Pentominoes (scale 3) rotated by four angles and uppercase (A-D) are four rotated large Pentominoes (scale 4). The values in the confusion matrix are the normalized predicted probabilities of each transform Pentomino being particular Pentomino type, where white means low probability and black illustrates high probability. ....	56
5.7	Examples of 112 texture images from Brodatz Album. ....	60
5.8	(a) Examples from the generated dataset, where their foreground and background textures are from the same texture images. (b) Examples with different foreground and background textures that are hard to distinguish. Foreground and background textures are randomly selected with replacement and the shape within the image is also randomly chosen from circle, pentagon, square, triangle and quarter. Two textures are mixed with roughly the same area for each one. ....	61
5.9	The generation process of the new texture dataset. First, two textures are randomly picked from Brodatz Album and assigned as the foreground or background texture. The 128 square regions are then cropped from the resized 160 square area at random locations. The two texture images are rotated by four choices of angles. Finally, the two textures are mixed with roughly the same pixels in a random shape. ....	63
5.10	Model architecture for region labeling. ....	64
5.11	Model architecture for supervised region labeling model. ....	65
5.12	(a) Samples from generated dataset with the same foreground and background textures and their false-color images indicating the label with the highest probability at each pixel. (b) Samples and their false-color images with different foreground and background textures. ....	68
6.1	An example of cumulated probability distribution over 729 ( $27 \times 27$ ) pixels in the region. The x-axis represents 112 labels. The y-axis is the probability sum over all pixels in the region, where 729 is the maximum value when the probabilities of all pixels are 1. ....	73
6.2	(a)(c) are the examples of same and different foreground and background textures, respectively, in the generated texture dataset. (b)(d) are the false-color images indicating the label with the highest probability at each pixel. (e)(f) are the cumulated probability distribution over all pixels in the region. ....	74

6.3	Comparison between the uncertainty and sparsity score with 320 examples. Each example contains 10 pixels within a window, showing eight significant textures appearing within a window. The x-axis represents the sparsity score while the y-axis shows the uncertainty score. The red line indicates uncertainty = 0.5. Three examples, $P_1$ , $P_2$ , and $P_3$ are marked as red crosses in the image. Different colors and glyphs represent various numbers of textures appeared in a window. When the uncertainty scores are less than 0.5, samples with different numbers of significant textures appeared in a window can be clearly identified by their sparsity scores. For example, if the samples have only one significant texture appeared in a window, the sparsity scores fall between 0 and 0.15, samples with two significant textures have the sparsity scores between 0.15 and 0.28, and etc. ....	85
6.4	Comparison between entropy and the sparsity score with 100 examples. Each example contains 100 pixels within a window, including 4 cases of different number of significant textures appeared within a window. The x-axis represents the sparsity score while the y-axis shows entropy. If the probabilities for all pixels are varied, different number of significant textures can be distinguished by using the sparsity score nevertheless entropy cannot. ....	93
6.5	A cumulative probability distribution over all samples within a mini-batch with high coefficient for the distinction score. ....	95
6.6	Examples of the same and different textures for foreground and background with high coefficient on the distinction score. When the coefficient of the distinction score sets high, the cumulative probability distribution shows that most of the textures have been recognized within a mini-batch and the false-color images also reflect the effects. ....	96
6.7	Examples of same and different textures for foreground and background with high coefficient on the sparsity score. From the cumulative probability distribution, the number of the textures been recognized within a mini-batch decreases to one significant peak for the same texture and two significant peaks for different, when the sparsity score sets high. There are also fewer colors dominating in the false-color images. ....	96
6.8	A cumulative probability distribution over all samples within a mini-batch with lower coefficient for the distinction score. ....	97
6.9	Comparison between the distinction and sparsity score with different coefficient settings. Note that the x-axis is the mean sparsity score over all samples within a mini-batch. ....	98

## CHAPTER 1: INTRODUCTION

Histology is the study of anatomy achieved by examining tissue samples under a microscope (Kayser et al., 2002). It is an essential and ubiquitous technique in biology and it is considered the gold standard for clinical diagnosis of many diseases (e.g., cancer detection). It also serves as an auxiliary tool for the identification of functional maladies (Odet et al., 2015; He et al., 2010; Junqueira and Carneiro, 2005; Mills, 2012).

The recent introduction of automated tissue processing, including automatic scanning and imaging of slides, has dramatically increased the speed for producing histology slides in the lab (McCann, 2015). Moreover, modern digitizing methods offer an opportunity to improve histopathologic procedures and diagnoses. It allows pathologists to view and browse digital versions of slides conveniently on a computer rather than through a microscope. It also has made histology images available for analysis via computer vision methods, thus providing new techniques to aid pathologists to quantify and examine the slides (Litjens et al., 2016).

In spite of these advances, much of the routine analysis for histology images remains a manual endeavor which relies heavily on the expertise of medical experts for interpretation and understanding (Gurcan et al., 2009). Pathologists can leverage quantitative and categorical parameters, such as cell types, cell counts, shapes, and lengths, but their final determinations still often depend on a more global analysis. This overall manual assessment is labor-intensive.

Over the past decade, machine learning techniques, such as deep learning, have shown a lot of promise for extracting semantics from images in computer vision. Deep learning is a recent extension to classical neural networks where learning architectures are formed by composing multiple non-linear transformations, with the goal of yielding more abstract and ultimately more useful representations (Bengio et al., 2013; Krizhevsky et al., 2012).

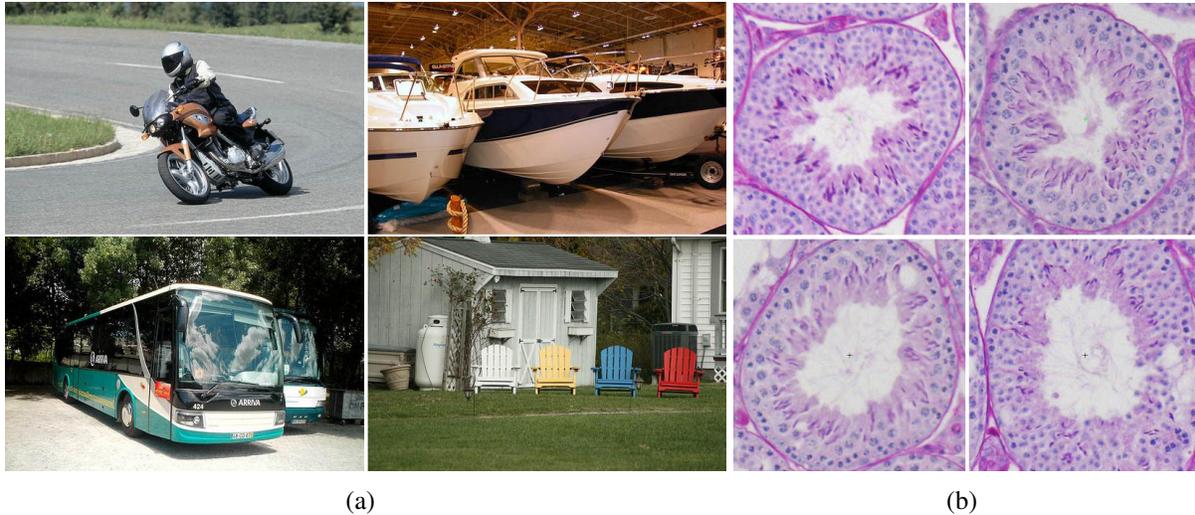


Figure 1.1: (a) Natural scene images from Pascal VOC 2012 (Everingham et al., 2011). (b) Stained histology images of normal/abnormal testis tubules. The objects in natural images generally appear at different depths, and are distinct from the background. In histology images, the tissue is at a single depth. Compared to the natural scene images, texture detailed information is essential to define an object.

## 1.1 Motivation

The naive application of standard deep learning approaches to solve problems in automated histology image analysis has, at best, met with limited success. One possible reason is that histology images are significantly different from natural images. Figure 1.1 points out some of the differences. In natural images, the colors are more varied and locally coherent making objects generally appear distinct from the background. Typical image analysis problems on natural images are like finding the bus in the image or classifying whether an object appears in the image.

On the other hand, histology represents a slice through a volume of tissue. Generally, they have particular structures with a few colors, and contain a wide variety of textures. An accurate assessment for automated histology analysis typically requires examining both the mix and configurations of different cell and tissue types, which locally appear normal and very similar in different states. Also, analyzing the histology images often involves the classification of tissue in different states, with similar textures and structure between states.

The notion of "normal" varies in different parts of a section, with different structures (blood vessels, etc) and different physiological states (for example, stages of germ cell differentiation in the testis and the arrangement of these stages within each tubule). A pathologist must understand the different aspects of normal histology to be able to discern abnormalities.

To predict the class for a given histology image, the learning model should first identify the regions where cells are abnormal or out of place. Applying these problems on natural images, it is more like trying to identify bulls in a herd of cattle. All these significant differences in histology images make transfer learning from natural images to histology images nearly impossible and also increase the difficulty for its classification.

Given these issues, I hypothesize that in order for a deep learning architecture for automated histology image analysis to capture high level semantic information, it must first learn small-scale image features, like color, shape, and texture. Deep learning is known for learning a hierarchy of filters that extract higher level of representations as the model goes up in the hierarchy. Features at the first few layers tend to pick out low-level properties of the input image like spatial frequency, texture, edges, or color. As the network goes up the hierarchy, the low-level properties start composing together, and become more recognizable objects like mouth, eyes, or noses. However, as the depth of neural network becomes deeper, a new research problem emerges. Because classical deep learning architectures only propagate the learned features from one previous layer, lots of valuable representations learned in the early layers are lost as they pass through layer by layer (He et al., 2016). This low-level information, considered as critical features for histology images analysis, are learned from the early layers in deep learning. Perhaps it would be valuable to preserve this information for subsequent layers.

In this dissertation, I propose two deep learning architectures to infer intermediate feature representations for histology image analysis. First, I proposed a novel deep learning architecture that introduces an intermediate layer to combine feature representations at different scales, under the assumption that if the decision layer (softmax layer) makes the decision based on the features learned at all scales, instead of the previous layer only, it might improve the performance. This

work inspired me that some low-level features the model identified correspond to the cell types at some level.

Taking one step further, I assumed that if I label the image with the probable cell type on a per pixel basis and feed the results to the traditional network, low-level representations can be preserved and the performance should be improved further as well. Recently researchers have proposed that, rather using deep-neural networks to solve difficult problems directly, it might be beneficial to decompose learning problems into smaller, intermediate tasks that provide hints for learning more complex concepts (Bengio et al., 2009b; Karpathy and Van De Panne, 2012). The paradigm of learning intermediate objectives for a high-level task, called *curriculum learning* (Bengio et al., 2009b; Kumar et al., 2010a; Lee and Grauman, 2011), is an emergent topic in deep learning and computer vision (Chen and Gupta, 2015; Dong et al., 2016; Gong et al., 2016; Pentina et al., 2015). Recent research (Hinton et al., 2006; Jiang et al., 2014) also shows that curriculum learning (Bengio et al., 2009b; Kumar et al., 2010b) can use an intermediate *hint* feature layer to steer difficult-to-learn objective functions towards effective solutions.

I therefore hypothesize that if the model can directly label the small-scale representations, like cell types, at each pixel or a given region in a histology image, and provide these labels as hints to the classifier about what possible tissue type the particular histology image would be, the model would learn better. I proposed a second deep architecture which applies the curriculum learning paradigm by introducing an intermediate learning task, learning a pixel/region labeling to provide hints to subsequent image classification task.

## **1.2 Thesis Statement**

By introducing an intermediate learning objectives in a deep learning network, one can improve classification accuracy of histology image analysis. The intermediate objectives act as hints or indirect supervision for higher level learning task.

For example, a pixel/region labeling may be a practical hint layer for deep learning. Intermediate objectives might also allow us to capture and preserve low-level representations, specifically texture

information. These types of labeling hints can generalize to other vision problems and be realized in either supervised or unsupervised settings.

### 1.3 Outline of Contributions

In defense of the thesis statement, this work explores how to construct a deep learning architecture with an informative intermediate layer. In completing the work, I have achieved the following goals:

**Using a Hyperlayer to Combine Features Across Scales.** A novel deep learning architecture for features learning and automated histology image analysis is presented. Using a stacked convolutional autoencoders, an intermediate layer, hyperlayer, is added between convolutional layers and hidden layers to collect the features from different scales. The work is described in Kao and McMillan (2017) and details are provided in Chapter 3.

**Pixel Labeling as an Intermediate Learning Objective.** I proposed a novel deep learning architecture for image classification that propagates specific domain knowledge from the pixel level to the image level in the spirit of curriculum learning. A labeling model is introduced to transform the information embedded in the image from color space to the probabilities of a small set of meaningful labels for each pixel. It serves as an intermediate hint, providing prior knowledge, for subsequent classification task. The details of the architecture are introduced in Kao et al. (2017) and Chapter 4.

**Generalizing Pixel Labeling Approach.** In order to demonstrate that pixel labeling model generalizes, I demonstrate it on two designed toy problems. One is the Pentomino problem, which is taken directly from the original curriculum learning paper, to evaluate whether the pixel labeling layer can accomplish a similar improvements to the Pentomino dataset learning task with the hint layers used in the original paper. The other toy problem is designed to demonstrate that pixel

labeling model is beneficial for classifying texture-like dataset. More details about the work are depicted in Kao et al. (2017) and Chapter 5.

**Unsupervised Pixel Labeling Layer.** I extended the labeling model to learn in an unsupervised fashion by introducing an unsupervised learning objective considering the correlation relationships among the labels. Three regularization terms, uncertainty, sparsity, and distinction, are designed to refine the label assignments by iteratively constructing an expected output distribution during the unsupervised labeling training. The work is detailed in Chapter 6.

**Conclusion.** Chapter 7 concludes this thesis and discusses potential areas for future research.

## **CHAPTER 2: BACKGROUND AND RELATED WORK**

### **2.1 Introduction**

Deep learning is a machine learning technique which has demonstrated notable successes in the last ten years (LeCun et al., 1998; Szegedy et al., 2015; He et al., 2016). Its outstanding ability to learn feature representations from data makes it the first choice for solving many machine learning problems. Deep learning is an extension of classical neural networks which utilizes multiple layers to learn important features and representations.

This chapter aims at presenting a broad overview of several common forms of neural network architectures, including the fundamentals of the networks and their use in computer vision tasks. The section explores the extensions of the neural network architectures. One is about the nature of convolutional neural networks, which has successfully been applied to analyzing visual imagery. The other is autoencoder, a special form of unsupervised learning of efficient codings. The chapter also outlines current research trends in curriculum based learning.

In addition, many deep learning techniques and applications have been described, as well as some work on automated histology analysis systems (Litjens et al., 2016; Cireşan et al., 2013, 2012). My work is at the intersection of these two areas. This chapter summarizes the development of automated histology image analysis systems and the previous work in deep learning, and concludes with a discussion of previous work on automated histology image analysis using deep learning.

### **2.2 Classical Neural Networks**

Neural networks are an information processing paradigm inspired by the brain's structure (McCulloch and Pitts, 1943; Widrow and Hoff, 1960; Rosenblatt, 1962; Rumelhart and McClelland, 1986). The basic unit in neural networks is a neuron, as shown in Figure 2.1. Neural networks are

composed of neurons connected by directed edges. An edge from node  $j$  to node  $i$  propagates the activation  $a_j$  from  $j$  to  $i$ . Each edge has a weight  $w_{ji}$  associated with it to indicate the strength and sign of the connection. Each neuron  $i$  computes a weighted sum of its inputs, and applies an activation function  $g$  to the sum, which derives the output  $a_i = g(\sum_{j=1}^d w_{ji}a_j)$ .

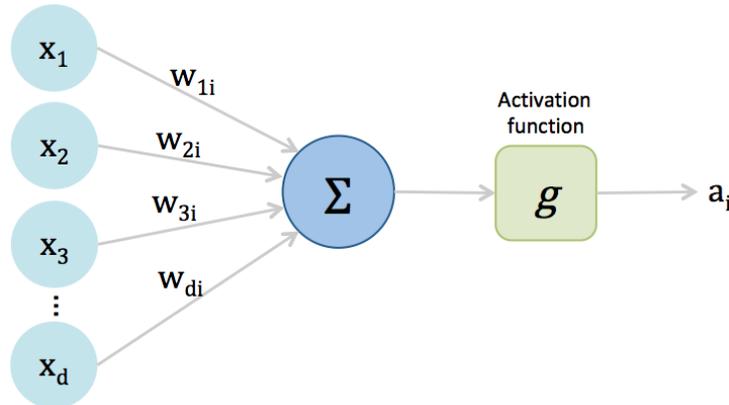


Figure 2.1: A sample unit  $i$  with input dimension  $d$ .

Neural networks are often modeled as distinct layers of neurons, called hidden layers. The neurons in neural networks only receive the input from the neurons in the immediately preceding layer. Given an input, the value will be propagated forward through the network layer by layer until reaching the output layer. This process of taking layers of features and feeding the features to another hidden layers is called feed-forward propagation.

Finally, the output is compared to the training data with an error evaluation function in the output layer. The errors and weights are used to tease apart which parts of the network contributed to misprediction and guide adjustment of the parameters in pursuit of better prediction. Neural networks can be trained to perform regression or classification tasks.

## 2.3 Deep Learning

Deep learning is a recent extension to classical neural networks. Deep learning architectures typically are formed by composing multiple non-linear transformations, with the goal of yielding more abstract and ultimately more useful representations (Bengio et al., 2009a). In recent years, deep

learning architectures have become popular, since they have outstanding performance in different computer vision and pattern recognition tasks (Hinton, 1990; Bengio et al., 2013; Krizhevsky et al., 2012; Farabet et al., 2013).

Convolutional neural networks have been especially successful in representation discovery. LeCun et al. proposed LeNet, which is used to read zip codes, digits, etc. AlexNet (Krizhevsky et al., 2012) is the next notable CNN used in computer vision, which has a very similar architecture to LeNet but has many more layers and parameters. Notably, a breakthrough in AlexNet is that multiple convolutional layers in AlexNet were stacked on top of each other whereas previous architectures included only a single convolutional layer.

VGGNet (Simonyan and Zisserman, 2014) demonstrated that for the task of image classification, deeper networks generally have better performance. There were 16 convolution and fully-connected layers in their best performing network. GoogleNet (Szegedy et al., 2015) proposed the concept of an inception module that dramatically reduces the number of parameters in the network. Also, instead of using fully-connected layers as in the upper layers of CNN, it uses average pooling to eliminate a large number of parameters.

Network depth is crucial in neural network architectures. Intuitively, deeper networks should perform no worse than their shallower counterparts. However, very deep networks degrade in performance. To address this issue, ResNet (He et al., 2016) bypassed signal from one layer to next via adding a shortcut connections with identity mapping, Wide residual network (Zagoruyko and Komodakis, 2016) decreased depth and increased width of residual networks, and ResNeXt (Xie et al., 2017) aggregated a set of transformations with the same topology.

### **2.3.1 Convolutional Neural Networks (CNN)**

CNNs are similar to classical feed-forward neural networks. Both learn sets of weights and biases, and each neuron receives inputs, performs a dot product, and applies a non-linearity to the result to generate an output. However, CNNs are very different from their special structures.

Unlike traditional neural networks, CNNs have repetitive blocks of neurons that are applied over each patch of the image. The weights for these repetitive blocks are shared. By using this special structure, CNNs are able to exploit spatial invariance in recognition. For example, a cat might appear many places in an image. For traditional neural networks, which do not have repeated weights across space, the group of neurons receiving inputs from the upper right corner of the image will have to learn to represent the cat independently from the group of neurons connected to the lower left corner. Meanwhile, we also need sufficient images of cats such that the network can train on several examples of cats at every possible image location separately. In addition, because the weights in CNNs are applied over each patch of the image, the training process will detect local patterns.

Given the context, a simple CNN is implemented as a repeated sequence of layers, including convolutional layers, pooling layers, and fully-connected layers (LeCun et al., 1989, 1998). The convolutional layer is a core building block in CNNs. A linear filter in convolutional layer is repeated across overlapping windows of the input signal, where all repeated units share the same parameters and form a feature map. To form a feature map, a function is repeated across subregions of the entire image, where the function  $h^k = \sigma(x * W^k + b^k)$  with parameters  $W$  and  $b$ .  $\sigma$  is an activation function, and  $*$  denotes convolution. Given a two-dimensional image  $x$  and a filter  $W$  of size  $h \times w$ , the convolution operator is computed by overlaying the filter on top of the image and summing the elementwise products between the image and the filter:  $(x * W^k)_{i,j} = \sum_{m=1}^h \sum_{n=1}^w x_{i+m-1,j+n-1} W_{m,n}^k$ .

A convolutional filter provides strong response on short snippets of data that exhibits a specific short pattern. In contrast to traditional computer vision, these patterns need not to be decided beforehand but can be learned from the data. Owing to the short length of the pattern, a convolutional filter has a small number of parameters. Outputs of multiple convolutional filters can themselves be processed by another set of convolutional filters. These higher level convolutional filters tend to detect larger patterns in the original image.

Another important concept in CNNs is a pooling layer, which is a form of non-linear down-sampling. The pooling layer partitions the convolutional outputs into disjoint regions and outputs a single summary statistic (frequently the maximum value, in which case it is called “max-pooling”) over these regions to obtain the pooled convolved features. The use of pooling is to decrease the spatial size of the representation and reduce computation in subsequent layers.

The lower levels of CNNs are composed of alternating convolutional layers and pooling layers. The upper layers are generally fully-connected layers as in traditional hidden layers models.

Compared to other computer vision algorithms, CNNs use relatively little pre-processing. CNNs are responsible for learning spatial filters that are frequently hand-engineered in traditional algorithms. Therefore, a CNN’s major advantage is that it is less dependent on prior knowledge and human effort in selecting features. CNNs can also learn general features. For example, they can be trained on one classification database, and then be applied to different tasks, so the need for learning task-specific features is minimized. Another feature of CNNs is that they often learn a hierarchy of filters, extracting higher level of representations as the model goes up the hierarchy. Filters in the first few layers tend to pick out low-level properties of the input image like edges, or color combinations. As the network progresses up the hierarchy, more complicated filters begin to emerge.

### **2.3.2 Autoencoder**

An autoencoder is an unsupervised learning algorithm that is used for dimensionality reduction and feature discovery. An autoencoder is a feed-forward neural network that attempts to discover a set of features that are able to reconstruct its input (Bengio et al., 2007; Poultney et al., 2007). An autoencoder maps an input  $x$  to an output (reconstruction)  $r$  through an internal representation or code  $h$ . The model consists of two parts: an encoder that tries to find the code to represent the input with the function  $h = f_{\theta}(x) = \sigma(Wx + b)$  with parameter  $\theta = \{W, b\}$ , and a decoder that produces a reconstruction with function  $\tilde{x} = f_{\theta'}(h) = \sigma(W'h + c)$  with  $\theta' = \{W', c\}$ . Two

parameter sets are usually constrained to be of the form  $W' = W^T$ , using the same weights for encoding the input and decoding the latent representation.

The reconstruction error can be measured in many ways, depending on the assumptions on the input distribution. As an example, the squared error  $\|x - \tilde{x}\|^2$  is a commonly used error metric. The objective is to make the code  $h$  capture the main factors of variation in the data. If the hidden layer is linear and the mean-squared error metric is used to train the network, the autoencoder would behave just like principal component analysis (PCA) (Karhunen and Joutsensalo, 1995). On the other hand, using a non-linear hidden layer, one can achieve encodings that differ from PCA, and are able to capture different aspects of the input distribution (Japkowicz et al., 2000).

Denosing autoencoders are a very common class of non-linear autoencoders where the autoencoder attempts to reconstruct the input from a corrupted version of it (Vincent et al., 2010). The idea is to force the hidden layer to discover more robust features and prevent it from learning the identity mapping. The corruption process is stochastic, and in one implementation it simply randomly assigns a subset of the inputs to zero. Then, the denosing autoencoder tries to predict the corrupted values from the uncorrupted ones.

### **2.3.3 Curriculum Learning**

Curriculum learning is a machine learning technique inspired by the way humans and animals learn. The idea of curriculum learning is that the learning process begins by learning simple concepts and progresses through information with increasing difficulties to learn more complex concepts (Bengio et al., 2009b). Gülçehre and Bengio (Gülçehre and Bengio, 2016) state that learning deep architectures is easier when some hints or indirect supervision are given about the function that the intermediate levels should complete. They also demonstrate that introducing an intermediate layer affects both the speed of convergence of the training process, and decrease the generalization error.

## 2.4 Automated Histology Image Systems

There has been considerable work on applying classification methods for analyzing histology data (Sirinukunwattana et al., 2016). The research on automated histology analysis systems focuses on analyzing the tissue as either an entire sample or at a cellular level (Arif and Rajpoot, 2007; Zhong et al., 2017; Chang et al., 2013). The approaches at the cellular level attempt to detect different cell classes based on the shape and texture of the nuclei.

Techniques from machine learning, such as  $k$ -nearest neighbor, support vector machines, genetic algorithms, etc. have been used to extract such features in order to to classify normal and abnormal tissues (Hattel et al., 2013; Nateghi et al., 2014; Kather et al., 2016; Zohra and Mohamed, 2009; Lannin et al., 2016). A few examples include the classification of Circulating Tumor Cells via random forests and support vector machines (Svensson et al., 2015), classification of diagnosing abnormal lung sounds using  $k$ -nearest neighbor (Chen et al., 2015), on-line monitoring of yeast cell density and viability via support vector machines (Wei et al., 2007), quantification of textural differences between cancer cells and blood cells (Phillips et al., 2012b,a), and classification of colonic tissue by means of a genetic algorithm (Amin et al., 2003).

Abnormalities can cause visible changes in the texture along with discernible changes in spatial arrangement and differences in cell shapes. Dalle et al. (2009) and Cosatto et al. (2008) exploited the properties of nuclei, such as shape, texture, and size, for grading in breast cancer images. To train a classifier for mitotic and non-mitotic cells, Malon and Cosatto (2013) used a combination of color, texture, and shape. Based on the appearance of the tumor and normal nuclei, Nguyen et al. (2011) were able to classify nuclei and identify glands for screening prostate cancer.

## 2.5 Deep Learning on Automated Histology Image Systems

Recent studies have shown that deep learning methods show promise for analyzing histopathological image datasets (Sirinukunwattana et al., 2016). Hafemann et al. (2014) demonstrated that a CNN is capable of surpassing traditional classification approaches. Wang et al. (2014a) was able to

classify mitotic cells by using a cascaded classifier trained on a combination of hand-crafted features and features learned through CNN. Wang et al. (2014b) expended the work on mitosis detection by combining handcrafted and convolutional neural network features. Other applications of CNN include learning a hierarchical part-based representation for breast cancer detection (Cruz-Roa et al., 2014), automated grading of gliomas (Ertosun and Rubin, 2015), and epithelium and stroma segmentation and classification (Xu et al., 2016a). Last, Xu et al. (2016b) used a stacked sparse autoencoder to learn a high level representation of nuclear and non-nuclear objects.

Typical histology image classification methods focus on either detecting cell types (Svensson et al., 2015; Wei et al., 2007), or classifying images in their entirety (Amin et al., 2003). The task for detecting cell types attempts to learn the features from local low-level information, like color, and classify different types of cells based on learned features. Since cell-type detection focuses on a small local area, it is often not sufficient enough to infer larger scale structural anomalies.

However, the results can be regarded as meaningful semantics for the cells within the surrounding structures, which is beneficial for classifying the larger structures. Most of the classification models distinguish abnormal structures by learning the features indirectly from low-level information (Dalle et al., 2009; Cosatto et al., 2008; Malon and Cosatto, 2013). The disadvantage of these models is that they are not able to directly recognize the cell types contributing to the abnormal structures since they only take low-level information into consideration. I believe that an intermediate labeling of low level structure, such as cell type, will significantly improve the performance of classifying abnormal structures.

## CHAPTER 3: USING A HYPERLAYER TO COMBINE FEATURES ACROSS SCALES

### 3.1 Introduction

Sperm concentration, morphology, and motility are common parameters measured in semen analysis for the clinical diagnosis of infertility (World Health Organization et al., 2010; Odet et al., 2015). However, a rich and complementary source of information for assessing male infertility is found via direct examination of histology images from testis cross sections. The mouse provides a powerful experimental model for addressing some limitations of human studies and for examining testis histology which helps unravel the underlying bimolecular causes of infertility.

A testis histology image can be segmented into units known as seminiferous tubules. The visual state of each tubule is indicative of many aspects of spermatogenesis, including the stage of spermatogenesis and structural abnormalities (Meistrich and Hess, 2013; Lanning et al., 2002; Oakberg, 1956). Figure 3.1 shows a representative testis histology, stained with Periodic acid–Schiff (PAS) and hematoxylin. This histology image can be decomposed into 248 tubules. Most tubules exhibit normal stages of spermatogenesis as shown in Figure 3.1. Even normal stages can vary with different structures and arrangements.

Figure 3.2 shows seminiferous tubules exhibiting defects, including vacuoles, abnormal germ cells, loss of germ cells, and the unexpected sloughing of materials into the normally clear lumen of the tubule. In many aspects, normal tubules look very similar to abnormal ones. For example, the visual appearances of a tubule with vacuoles might look normal if you examine it locally as shown in Figure 3.2(b). Also, local structures and cells might seem normal but appear in the wrong place like Figure 3.2(e). Classifying a tubule as normal or not needs to consider the overall context and different scales of cells. Because of this subtlety, significant training is required before a human observer can differentiate between them.

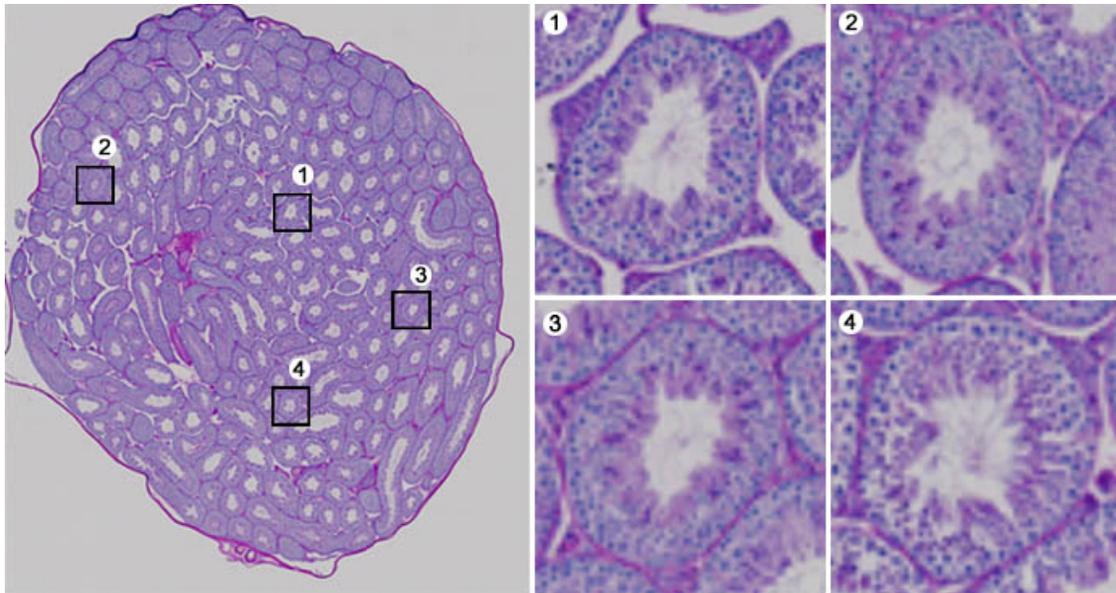


Figure 3.1: A composite testis image from lab mice with normal seminiferous tubules. Tubules 1, 2, 3, and 4 are shown at greater magnification.

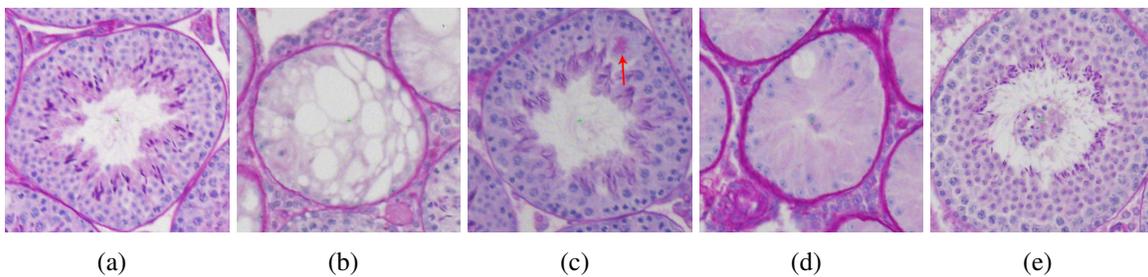


Figure 3.2: Examples of tubule with different abnormalities. (a) a normal tubule (b) a tubule with many vacuoles (c) a tubule with abnormal germ cells (red arrow) (d) tubule with complete germ cell loss and one large vacuole (few vacuoles in the manual labeling scheme) (e) a tubule with germ cells in lumen.

To solve this problem, I propose a novel deep learning architecture where an additional layer, called a *hyperlayer*, is created that combines with convolutional layers to capture high level semantic information and small scale image features. The proposed approach does not follow traditional convolutional network architecture where each layer's outputs feed only to the next and the top layer only is used for classification, which may be too coarse to reveal the nuance among different stages of tubules.

The deep learning architecture composes multiple linear and non-linear transformations of the data to infer a more abstract and useful representation (Cruz-Roa et al., 2013). The attraction of this

approach is that feature extraction is part of the learning process. My goal is to build a learning architecture which can extract the appropriate features, visualize the features learned by the model, and classify the tubules based on the features learned.

In the architecture, the learning network is able to capture features of an image from different levels of abstraction and scale and thus provide a more complete description for the classification process. Inspired by deep learning in image classification, the novel deep learning architecture combines convolutional autoencoders, a hyperlayer, and a softmax classifier for tubule classification and visual representation.

## **3.2 Approach**

The proposed architecture for testis histology image learning, visualization of learning process, feature combination, and classification is based on deep learning network architecture depicted in Figure 3.3(a). I employed stacked convolutional autoencoders (Masci et al., 2011) for classification, which includes unsupervised pre-training stage (green block) and supervised fine-tuning stage (blue block). In pre-training stage, the architecture was trained via convolutional autoencoders one layer at a time. Once all layers are pre-trained, the network goes through to the second fine-tuning stage, where I added the hyperlayer to capture features from all layers/scales from the first stage. As for visualization of learning process, I reconstructed the image with learned features via deconvolutional networks. Each module is corresponding to the layers in the architecture described in the following sections.

### **3.2.1 Feature Learning via Stacked Convolutional Autoencoder**

Convolutional autoencoder architectures are similar to standard autoencoders, except the weights in convolutional neurons are shared to achieve spatial/position independence (Masci et al., 2011). Several convolutional autoencoders can be stacked to form a deep learning architecture. Each layer takes the max-pooling of the hidden outputs from the layer below as its input, and performs unsupervised pre-training. The pre-training process first takes the input  $x$  and maps

it to the  $k$ -th feature map using the function  $h^k = \sigma(x * W^k + b^k)$  with parameters,  $W$  and  $b$ .  $\sigma$  is an activation function, and  $*$  denotes the 2D convolution. Afterward, the feature maps are used to reconstruct the input by mapping back with the function  $y = \sigma(\sum h^k * \tilde{W}^k + c)$  with parameters  $\tilde{W}$  and  $c$ . Those parameters are optimized by minimizing the differences between input and the reconstructed representation. The learned features from convolutional autoencoders are used to initialize a convolutional neural network with identical topology. Once the first stage unsupervised pre-training is done, the network goes through the convolutional neural network in the second fine-tuning stage as shown by the blue dash lines in Figure 3.3(a).

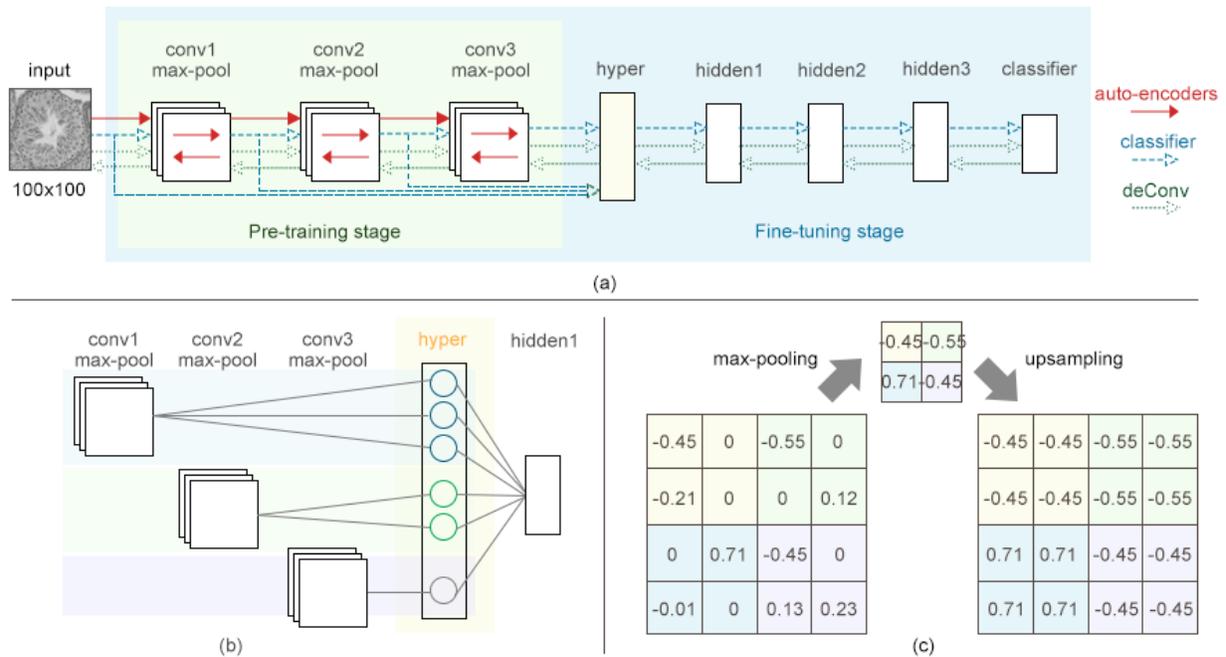


Figure 3.3: (a) Overview of our deep learning architecture, where the building blocks within green box are convolutional auto-encoders performed in the first stage of classification, and all building blocks within blue box compose a whole learning framework for the fine-tuning stage. Deconvolutional networks indicated in green dotted lines go through whole framework to reconstruct the image from the top most layer. (b) Example of a hyper layer with weights 3, 2, and 1 on convolutional layer 1, 2, and 3, respectively. (c) Example of upsampling operation on pool size  $2 \times 2$  in deconvolutional networks. Pool-sized neighborhoods are filled with the absolute max value within that area.

### 3.2.2 Hyperlayer construction

In the fine-tuning stage, unlike a traditional stacked autoencoder architecture, I added an intermediate layer, called a hyperlayer, between convolutional layers and hidden layers to collect features from different layers. When pooling is applied, these features can be considered different scales. A hyperlayer is a fully connected layer which concatenates extracted feature maps from every convolutional layer below and outputs the representative neurons of the corresponding layer to above layer. In addition, hyperlayer is allowed to specify the importance of every convolutional layer by configuring the weights for each of them. Figure 3.3(b) illustrates an example of the hyperlayer with weights 3, 2, and 1 on features collected from convolutional layer 1, 2, and 3, respectively. Combining all features across multiple layers provides a representation with features derived from different scales of details to the classifier layer.

### 3.2.3 Automated Classification via Softmax Classifier

Beyond the hyper layer, I constructed three hidden layers to decrease the number of neurons in the framework and build a logistic regression classifier in the end of the architecture. The classifier was trained by minimizing the negative log-likelihood. The output of the classifier is the probability that the input  $x$  is a member of a class  $i$ , and then I chose the class whose probability is maximal as our model's prediction.

### 3.2.4 Image Representation via Deconvolutional Networks

I used convolutional autoencoders to learn the features that can best reconstruct their input in a layer-wise fashion. Therefore, the reconstructed image reflects the features learned from the model in layers. To see the features learned from the whole framework including classification, I exploited deconvolutional networks (Zeiler et al., 2011) to reconstruct the image from the classifier layer. The concept of the deconvolutional network is very similar to convolutional autoencoders, except for the max-pooling part. In deconvolutional network, to reconstruct the image I have to upsample the results of max-pooling to form the hidden representation and then I can reconstruct the input, while

autoencoders can directly take the hidden representation to reconstruct the input. In the model, upsampling is operated by using the max value in the pool-sized for all neighborhoods. An example of upsampling is depicted in Figure 3.3(c).

### **3.3 Experimental Setup**

In this section, I introduce a histology image dataset, that the hyperlayer is applied to, and show how I conducted the experiments on this dataset. Here I also present the architecture for the hyperlayer classifier, including the number of layers, the type of the layers, the size for the filters, and the weights for the convolutional layers.

#### **3.3.1 Histology Image Dataset**

The approach is evaluated on the histology images of mice, which are collected from a panel of multiparental lines, called Collaborative Cross (CC), from five classical inbred strains (A/J, C57BL/6J, 129S1/SvImJ, NOD/ShiLtJ, and NZO/HILtJ) and three wild-derived strains (CAST/EiJ, PWK/PhJ, and WSB/EiJ). During the generation of the CC mice, a large portion of the CC lines stopped producing offspring leading to a high extinction rate (Odet et al., 2015; Collaborative Cross Consortium et al., 2012). Odet et al. (2015) observed that the large fraction of extinction was due to male infertility. To facilitate a better understanding of varied breeding performance in the CC, a separate study of each of the eight founder strains of the CC was undertaken (Odet et al., 2015; Shorter et al., 2017).

The histology images from male breeders were stained with PAS and hematoxylin, and scanned with 20x magnification. A custom interactive image analysis tool was developed for annotating the digitized histology images. The tool automatically identifies seminiferous tubule centers based on a random-forest classifier (Odet et al., 2015). We used the interactive tool to annotate the classes of each tubule, and the final annotation information was kept in a database. From 1,417 testis histology images, 52,122 tubule image patches of size  $100 \times 100$  were extracted for the tubule classification.

The dataset contains five categories: normal seminiferous tubules, tubules with few/many vacuoles, tubules with abnormal germ cells, tubules with germ cells in the lumen, and tubules germ cell loss. Examples of tubules in each of these five categories are shown in Figure 3.2. The five categories are not mutually exclusive, and non-trivial to classify. The images were manually annotated by an expert in male reproduction, and their status was labeled in each of these five categories. I conducted two experiments on individual subset of the dataset, each of them has 10,000 tubule sub-images in total. To increase the statistical power, I replicated some classes of abnormal tubules by rotating them into one of four different angles, and randomly picked equal amount of images for each class to keep the dataset unbiased. The first experiment took the images from normal tubules without germ cell loss and tubules with many vacuoles with various degrees of germ cell loss. 5,000 images were used for both classes. The second one uses the subset of normal tubules, tubules with many vacuoles and various degrees of germ cell loss, tubules with the last stage of germ cell loss, and tubules with both many vacuoles and the last stage of germ cell loss, 2,500 images for each of them.

### **3.3.2 Building the Classifier**

I partitioned the images into three subsets: training dataset, test dataset, and validation dataset. The datasets was randomly partitioned into 6 equal sized subsamples. Validation and testing datasets each used a partition, and the remaining 4 were used as training data. I trained the approach on data in training dataset, and classified the tubules in test dataset. Hyperparameter values were optimized by random search. I randomly selected the number of hidden units from [50, 100, 200, 300, 500, 1024, 2048] and randomly selected the learning rates within the interval of [0.001, 0.5]. As for the classifier, I had three convolutional autoencoders layers, one hyperlayer, and three fully-connected hidden layers. The combination of optimized hyperparameters is listed in Table 3.1. I normalized all images so that their red, green, and blue channels fell between -1 and 1, and used hyperbolic tangent as the activation function. This normalization maintains a consistent input range for all layers. Note

Table 3.1: The hyperparameter settings for all layers in the learning architecture. Layer type: I - input, C - convolutional layer, MP - max-pooling layer, H - hyper layer, FC - fully-connected hidden layer, LG - logistic regression layer.

Layer	Type	Maps	Neurons	Filter size	Weights
0	I	1	100×100	-	-
1	C	50	50×50	13×13	50
2	MP	50	25×25	2×2	-
3	C	100	25×25	9×9	100
4	MP	100	13×13	2×2	-
5	C	150	13×13	3×3	150
6	MP	150	7×7	2×2	-
7	H		900	-	-
8	FC		512	-	-
9	FC		100	-	-
10	FC		50	-	-
11	LG		2	-	-

that when I convolved the input, the surrounding missing part are padded symmetrically with the input to keep the dimension of output image equivalent to the input.

### 3.3.3 Classification Performance and Comparison

To evaluate the experiment results, I compared the approach against traditional CNN and standard stacked convolutional autoencoders (which does not include the hyperlayer) with same hyperparameters.

## 3.4 Experimental Results

I present the results of applying hyperlayer on a histology image dataset in this section. The results indicates that hyperlayer can effectively improve the accuracy, precision and recall, and specificity compared to traditional CNN.

Table 3.2: Comparison of classification performance on the first dataset (normal / many vacuoles), where SCAE is Stacked Convolutional AutoEncoders. I compared the method with traditional CNN and SCAE with same parameters. The best results are in bold typeface. Accuracy is the ratio of correctly predicted observation to the total observations. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. Recall measures the proportion of positives that are correctly identified, while specificity measures the proportion of negatives that are correctly identified.

	Accuracy	Precision	Recall	Specificity
CNN	0.901	0.348	0.862	0.904
SCAE	0.973	0.981	0.964	0.982
Proposed methods	<b>0.986</b>	<b>0.987</b>	<b>0.984</b>	<b>0.988</b>

Table 3.3: Comparison of classification performance on the second dataset (normal / many vacuoles / last stage of germ cell loss / many vacuoles and last stage of germ cell loss). The hyperparameters are same in these three methods. The best results are in bold typeface.

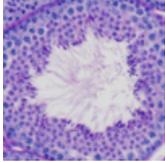
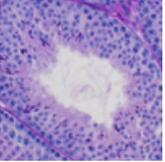
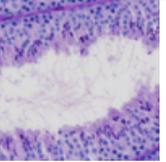
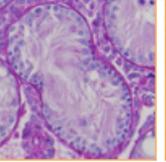
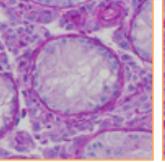
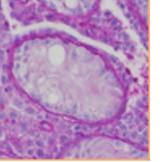
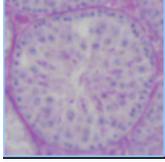
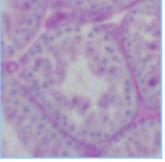
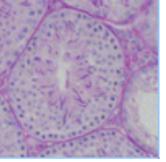
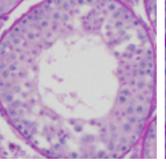
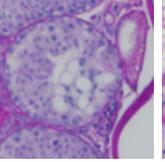
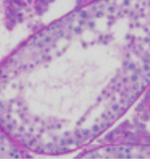
	CNN	SCAE	Proposed methods
Error rate	5.269	3.533	<b>2.150</b>

### 3.4.1 Automatic Classification Performance

Table 3.2 shows the classification performance results in terms of accuracy, precision, recall, and specificity on the first dataset. Table 3.3 compares the error rate among traditional CNN, SCAE, and our approach on the second dataset. The results show that incorporating a hyperlayer outperforms the canonical methods on a testis histology dataset. This suggests that the proposed architecture has captured and learned the important features. Including a hyperlayer preserves the details at the lower layer which are critical in our dataset, while the traditional CNN or standard stacked convolutional autoencoders method loses information as data propagates to the end of the learning process.

Table 3.4 illustrates the examples of classification results in the first experiment. For each, I report the image patch, corresponding prediction result and the ground truth. For a subset of normal tubules I misclassified them as tubules with vacuoles, as well as the other way around, these misclassifications, when examined manually are extremely difficult to determine whether they are

Table 3.4: Examples of our models classification results compared to the ground truth. The normal tubules that the model identified with vacuoles are arguable since most appear to exhibit vacuole-like features.

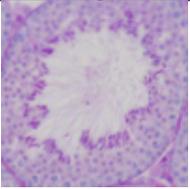
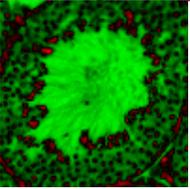
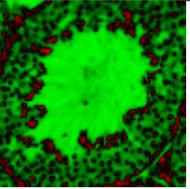
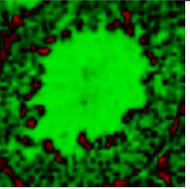
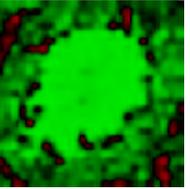
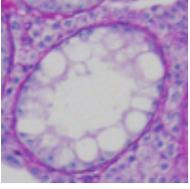
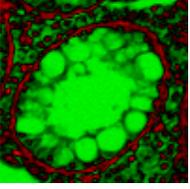
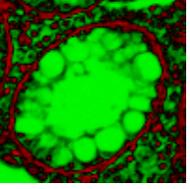
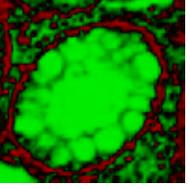
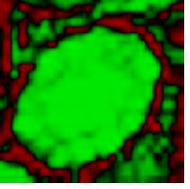
	Normal (True)			Vacuoles (True)		
Normal (Predicted)						
Vacuoles (Predicted)						

normal tubules or tubules with vacuoles. They are in essence ambiguous in terms of vacuoles. Even for biologists with expertise in this area, they probably would not always reach a consistent classification result.

### 3.4.2 Image Reconstruction with Deconvolutional Networks

Table 3.5 shows the results of image reconstructions via deconvolutional networks. I normalized the images before learning the features to alleviate the color variations stemming from hematoxylin staining. I highlight the values below zero in red and above in green. Because of max-pooling following by the convolutional layer, the resolution of the images decreases as the layer goes up. In addition to the max-pooling layer, convolution operation smooths the image while applying the filter over the whole image. Therefore, I observed that the reconstructed image became blurrier at the higher layer as expected. The reconstructed image generated via deconvolutional networks takes the weight after fine-tuning stage and propagates the results through the bottom layer. I adopt the tuned weights to reconstruct the image, and hence, the image reflects the classification features learned from the model.

Table 3.5: Examples of reconstructed images generated from the deconvolutional networks by taking the hidden representation of the particular layer and cascading it back to the original image. Notice how the reconstructed image becomes blurrier at higher layer because max-pooling and convolution tend to decrease the resolution and smooth the images at higher layers.

	Original Image	Norm Image	Layer 1	Layer 2	Layer 3
Normal					
Vacuoles					

### 3.5 Conclusion

I present a novel deep learning architecture for features learning and automated tubule classification for testis histology images. Based on stacked convolutional autoencoders, I insert a hyperlayer, collecting features from every convolutional layer below to the classifier. The approach outperforms canonical deep networks and shows an improvement over the existing methods. Plus, I construct an extension of the architecture to see how well the features learned by reconstructing the images from top to bottom. The work helps provide a better understanding of the prediction produced by the automated classifier.

Although the proposed architecture has exhibited the improvement on the classification problem, it may show concerns when the networks go deeper. Because the hyperlayer collects the learned representations from all convolutional layers above, the architecture does not scale very well. When the architecture becomes deeper, the inputs from all convolutional layers might become overwhelming. This might increase the training time and the need for immense amount of available memory during training. Since not all representations are equally important to the task, one solution might combine the critical representations from one or few convolutional layers instead. In this case,

the model is able to preserve the important information from the early layers and the scalability would not be an issue anymore.

The other disadvantage of the proposed architecture is that I use autoencoder to learn the features in the unsupervised pretraining process. The features learned from autoencoder may not meaningful semantics since the unsupervised learning process of autoencoder is optimized by minimizing the difference between the input and the reconstructed output. Thus, the nature of autoencoder, the learned features may not be specific enough to help classify the normal tubules in different stages. To address the problem, the concept of combining low-level representations, texture particularly, and learning meaningful semantics, from a single convolutional layer is explored further in Chapter 4.

## CHAPTER 4: PIXEL LABELING AS AN INTERMEDIATE LEARNING OBJECTIVE

### 4.1 Introduction

In Chapter 3, I presented a deep learning architecture with an additional intermediate layer, called a hyperlayer, to preserve low-level representations. This addition proved to be beneficial to the learning task in that it improved the classification accuracy of tubules from histology images. This suggests that some of the low-level representations the model identified might correspond to the cell types in the histology images, which is a common approach used by pathologist when assessing histology images.

Figure 4.1 shows the examples of normal and abnormal tubule histology images and corresponding false-color images with the different colors illustrating regions composed of different cell types at each pixel. For these two examples, the original histology images have similar colors, shapes, and textures, but the cell types are significantly different. I therefore hypothesize that instead of combining the low-level representations to the model, if the model can infer cell types information based on the low-level representations to the particular deep learning architecture, it should be more informative and useful.

Meanwhile, recent research (Hinton et al., 2006; Jiang et al., 2014) shows that curriculum learning (Bengio et al., 2009b; Kumar et al., 2010b) can use an intermediate *hint* feature layer to steer difficult to learn objective functions towards effective solutions. This motivated me to explore combining the cell-type information as an intermediate hint layer in an effort to improve the classification performance.

Given local context, a per pixel annotation might help address the histology classification problem. Per pixel annotations are commonly used in medical-image-analysis tasks (Gurcan et al., 2009; Massar et al., 2011; Riordan et al., 2015) including feature extraction (Boucheron, 2008;

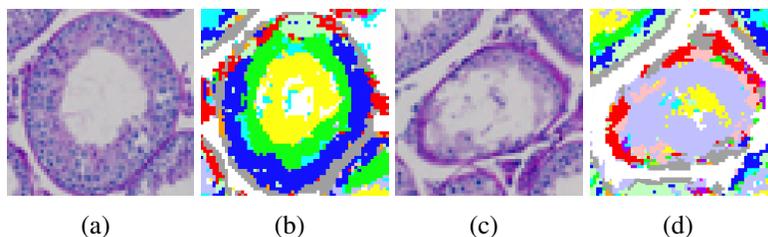


Figure 4.1: (a)(c) Examples of true-color testis histology cross sections of normal and abnormal tubules. (b)(d) False-color images indicating possible cell types at each pixel.

Gil et al., 2002; Sims et al., 2003), image segmentation (Boucheron et al., 2007; Can et al., 2008; Fernandez et al., 2005), and classification (Doyle et al., 2006). The joint analysis of individually labeled pixels has also been shown to aid in summarizing entire regions of an image (Massar et al., 2011).

Therefore, I propose to apply the curriculum learning paradigm by introducing an intermediate learning task that attempt to learn a supervised *pixel labeling* to provide hints to subsequent image classification layers. This architecture requires additional annotations beyond the larger *meta-classification* task. Specifically, it will require the selection of labels that partitions pixels into a small number of classes. Instead of directly classifying the images, the model first learns a label for the center pixel given the context of its neighbors. The image classifier then uses these labels to classify the images. I demonstrate my approach on a task of detecting abnormalities in medical histology images. In my application, I use region, cell, and tissue type as intermediate pixel labels.

The main contributions of this work are a novel deep learning architecture for image classification that propagates specific domain knowledge from the pixel level to the image level in the spirit of curriculum learning. Extensive experiments show that our architecture outperforms other traditional deep-learning approaches.

## 4.2 Approach

My goal is to learn intermediate representations that can improve the precision and recall of an image classifier. Inspired by curriculum learning (Gülçehre and Bengio, 2016), I trained an

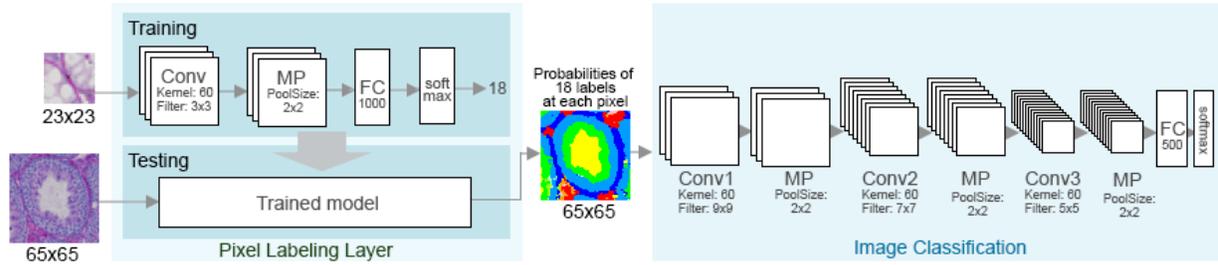


Figure 4.2: The classification task is divided into two subtasks, a per-pixel labeling and an image classification. The pixel labeling model is trained using a CNN trained on  $23 \times 23$  image patches, whose output is a likelihood function of 18 possible labels. Once learned, this model is applied to every pixel of an image ( $65 \times 65$ ) and then used as input to train the image classifier.

intermediate layer to infer class labels for every pixel based on textures and structure, which are then used as a hint for our meta task, image classification. This effectively divides our classifier into two learning subtasks, a per-pixel labeling and an image-level classification as shown in Figure 4.2.

The first learning task infers a per-pixel label from a fixed palette. The pixel-labels are learned from small patches surrounding a central pixel and are separately annotated by a domain expert. The deep-learning architecture transforms the three-channel pixel intensities into label probabilities for meaningful classes, which, in the histology domain, includes cell and tissue types, cell layers, and various classes of empty regions. After the pixel labeling model is learned, the model is then applied to predict all images in the image classification dataset. Finally, an optimized image classification model can be learned based on the hints provided by the intermediate pixel labeling layer. The details of the pixel labeling model and image classification model are described in the following sections.

#### 4.2.1 Pixel Labeling Layer

In the pixel labeling layer, I predefined a set of labels for identifying the pixels in an image, and selected the window size of the neighboring pixels for constructing a feature vector at the given pixel. The goal of this intermediate layer is to categorize the pixels in an image into a small number of defined labels and output the distributions of label probabilities at each pixel in an image.

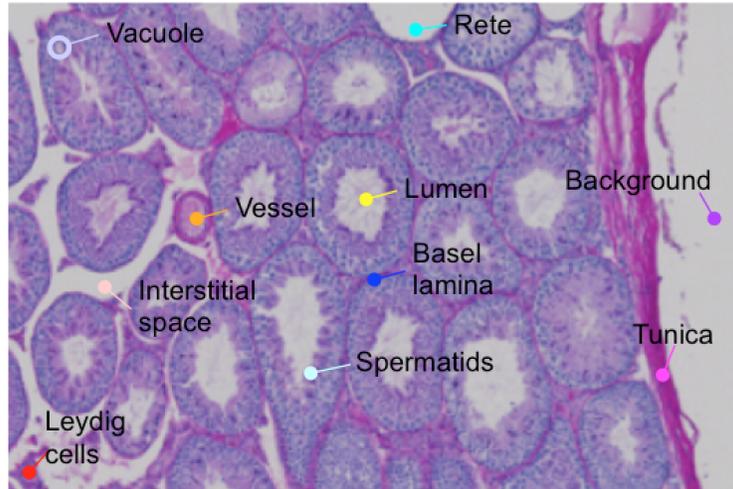


Figure 4.3: Supervised labels are predefined for characterizing each pixel. The labels define the structure of a testis from outside of the testis to the center of the tubule, varied from different kinds of empty space to specific cell types.

#### 4.2.1.1 Supervised Pixel Labels

I selected 18 supervised labels with domain specific meanings, and hand labeled random uniformly spatial patches from the image training dataset by an expert. Each was visually distinct and commonly observed in testis cross sections. As illustrated in Figure 4.3, they included background regions outside of the cross section, tunica, interstitial space, Leydig cells, blood vessel, basal lamina, Sertoli cytoplasm, spermatogonia and early-stage spermatocytes, late-stage spermatocytes, early-stage spermatids, late-stage spermatids, seminiferous tubule lumen, rete testis, PAS accumulations, tumors, cells in lumen, vacuoles, and other. The image patches are largely not contained within the training images used for the tubule classification. Each patch indicates the cell-type, tissue-type, anatomical region, background, or staining artifacts of the center pixel.

#### 4.2.1.2 Window Selection

Pixel labeling assigns a vector of label likelihoods to the center pixel given the intensities of its neighborhood as context. I selected size window centered at the given pixel and the pixels inside the window are regarded as neighbors. With regard to selecting the window size, I discovered that if it is too big, the window contains lots of neighbors which are irrelevant to the center pixel. On the

other hand, if the window size was too small, the information is not sufficient to accurately label the center pixel. In this paper, I chose square window of size  $N \times N$ , centered at the given pixel. To decide the optimal  $N$ , I exhaustively tested on the validation dataset and choose the one with the best performance. I found that the window size of 23 performs best in the experiments since it is not too large to contain the whole tissue and not too small that cannot distinguish the small cell type.

#### 4.2.1.3 Data Normalization

It is important to normalize the intensities of data so that the data dimensions are of approximately the same scale. Many researchers have been proposed solutions to the normalization problems in computer vision and deep learning (Ioffe and Szegedy, 2015; Shimodaira, 2000). In this section, I opted for an approach that maps the histogram distribution of all pixel intensities in the training and testing dataset to a template histogram distribution, where the highest and lowest value of three color channels are the same. I tried standard quantile normalization, and it did not show significant difference but more computational efforts. Extensive experiments with different template histogram distributions suggest that normalization is crucial to the proposed approach, since the representations with similar texture but different colors might indicate different cell types.

#### 4.2.1.4 Pixel Labeling Model

I used a standard CNN with a single convolutional layer, a max-pooling layer, whose outputs formed a hidden layer feeding a fully-connected second layer with a softmax nonlinearity as shown in Table 4.1. The convolutional layer learned coefficients for 60  $3 \times 3$  kernels with a stride of one pixel. Max-pooling was used to forward one feature from each  $2 \times 2$  neighborhood to the next level, 1,000 hidden units, after applying a 30% dropout rate. The detailed configuration of the pixel labeling model are also listed in Table 4.1. The resulting model takes the input patch  $p_i$  and maps it to the  $k$ -th feature map using the following function:

$$h^k = \text{maxpool}(\sigma(\mathbf{W}^k * p_i + b^k)) \quad (4.1)$$

Table 4.1: The configuration details of our two-tiered architecture, pixel labeling model and image classification model. Layer type: Conv - convolutional layer, FC - fully-connected layer.

Pixel Labeling	Image Classification
Input 23×23	Input 65×65
Cov-60 3×3	Cov-60 9×9
Maxpooling 2×2	
	Conv-60 7×7
	Maxpooling 2×2
	Conv-60 5×5
	Maxpooling 2×2
FC-1000	FC-500
softmax	

where  $W$  and  $b$  are weights and bias, and  $*$  denotes the 2D convolution.  $\sigma$  is a point-wise nonlinearity hyperbolic tangent function, while  $\text{maxpool}$  is a function that partitions the convolutional outputs into regions and outputs a maximum value for each subregion. After a fully-connected layer, the output of patch  $p_i$  is defined as follows:

$$p_i = \sigma(\mathbf{UH} + c) \quad (4.2)$$

where  $H$  is the flattened  $k$ -feature maps, and  $U$  and  $c$  are weights and bias. I used hyperbolic tangent for activation function  $\sigma$ .

I trained the model with the intermediate target  $Y$ . Let  $l_i$  be the normalized prediction vector from the classifier for pixel  $i$ , and  $L$  is the number of labels defined in pixel labeling layer. I utilized the softmax loss function by computing normalized predicted probability distributions over labels  $l_{i,y}$ .

$$l_{i,y} = p(l_i = y_i | p_i) = \frac{\exp(\mathbf{W}_y^T p_i)}{\sum_{k=1}^{|L|} \exp(\mathbf{W}_k^T p_i)} \quad (4.3)$$

where  $W$  refers to the corresponding prediction function parameter. For the overall loss on a patch  $p_i$ , the model learns the parameters by minimizing the negative log-likelihood given all the labels in the training dataset  $D$ :

$$loss_{p_i} = - \sum_{i=0}^{|D|} \log P(l_i = y_i | p_i) \quad (4.4)$$

Finally, when the pixel labeling model is trained, the predicted label probability  $l_{i,y}$  for patch  $p_i$ , which will be propagated to subsequent layers, can be either presented as a probability distribution of labels from softmax output at pixel  $i$  or a hard target vector with  $l_{i,y} = 1$  if pixel  $i$  is labeled as  $y_i$ , and 0 otherwise. For training maximally discriminative features for the image classification, I chose to use a probability distribution of labels in the pixel labeling layer.

#### 4.2.2 Image Reference

Once pixel labels are learned, I applied them to all images in the image classification dataset. The model converts three-channeled pixel intensities to label probabilities to serve as hints for the image classification. Pixel labeling model only needs to be trained on a sparse data, but it predicts dense label at every pixel for the subsequent image classification task.

#### 4.2.3 Image Classification

For image classification, I aim to obtain an optimized image classification model based on the hints provided by intermediate pixel labeling layer. This is achieved by learning a CNN with probability distributions of meaningful labels on small cell type as input and output the predicted image class as the results. Table 4.1 shows the details of the configuration in the image classification model. I trained the parameters by minimizing the negative log-likelihood on target  $z$ , and computed the normalized predicted probability distributions for every image  $x_j$  over classes  $c_{j,z}$ :

$$c_{j,z} = p(c_j = z_j | x_j) = \frac{\exp(\mathbf{W}_z^T x_j)}{\sum_{k=1}^{|C|} \exp(\mathbf{W}_k^T x_j)} \quad (4.5)$$

where  $W$  is a temporary weight matrix used to learn the features and  $|C|$  is the number of classes in classification model. For each image  $x_j$ , the class is produced by:

$$c_j = \arg \max_{z \in C} c_{j,z} \quad (4.6)$$

Finally, the predicted class of a given image is assigned with the argmax of the probability distribution of all classes.

### 4.3 Experiment Setup

In order to evaluate the effectiveness of my pixel labeling model on histology images, I applied it to a large dataset. This section shows how I prepared the datasets for both pixel labeling task and the meta classification task.

#### 4.3.1 Histology Image Dataset

I demonstrated the pixel labeling approach on testis histology images of mice. The histology images were stained with PAS and hematoxylin, scanned at 20x magnification, and reduced by 12.5% in both directions giving an effective magnification of 2.5x. A testis histology image was pre-segmented into units called seminiferous tubules. The first phase of the seminiferous tubule pre-segmentation process was the automatic off-line identification of seminiferous tubule centers based on a random-forest classifier that used local-color histograms at each pixel as features, details provided in the paper (Odet et al., 2015). A Web-based interactive tool was developed to correct both missed and mislabeled centers, and for experts to annotate features of each tubule, including abnormalities such as vacuoles and the loss of germ cells.

The dataset of the experiments here is composed of pre-segmented tubule images derived from the pre-segmentation process, and each tubule image sample is centered by the label point. Figure 3.2 shows samples of histology images of seminiferous tubules. The objective of the experiment was to classify the state of germ-cell loss and identify abnormalities in each tubule. This problem is

challenging because an accurate assessment requires examining both the mix and configurations of differing cell and tissue types, which locally appear normal. However, cells can be out of place, expected cell types might be missing, or the morphology of one-cell type might change due to the absence of others.

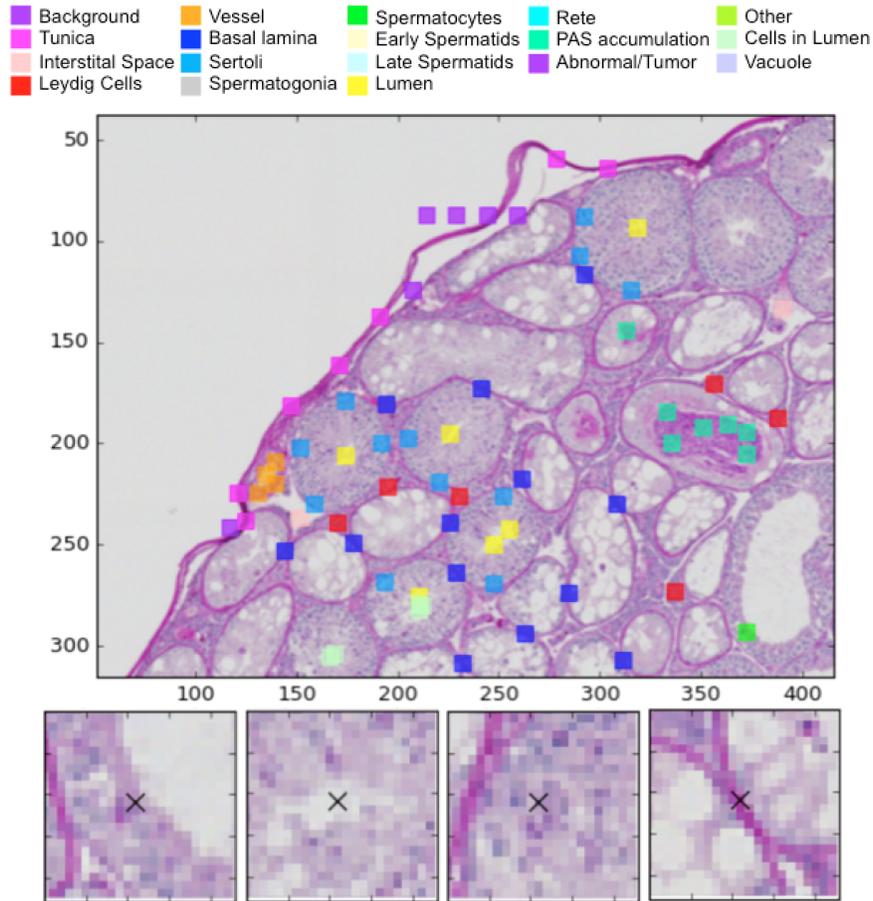


Figure 4.4: Above image is the screenshot of interactive tool designed for labeling patches. The squares with different colors represent 18 predefined labels. Images below are the patch examples in the labeling dataset, where black cross indicates the center of the tubule.

### 4.3.2 Pixel Labeling Layer

For the pixel labeling model, I collected a training dataset by randomly picking small  $23 \times 23$  patch pixels from 243 histology images with the objective of uniformly sampling each of the pixel-label classes with approximately 1,500 examples. Figure 4.4 illustrates the system designed for labeling the patches, and the patch examples in the pixel learning dataset as well. The patch

samples include multiple cells and other surrounding information, but generally contained less than 12% of a tubule. The label was assigned for the center pixel while the surrounding 528 pixels provided context. Human experts are able to perform this task, determining the cell, tissue, or region of origin for a  $23 \times 23$  patch, with ease.

The final dataset was composed of approximately 21,000 patches in total. Note that while 21,000 patches might seem large, consider that every histology image contains millions of patches, thus the training set is very sparse. The dataset was then partitioned into six equal-sized subsamples, where validation and testing datasets each used a partition, and the remaining 4 were used as training data.

### **4.3.3 Image Classification Layer**

For image classification, I used a set of tubule images of size  $65 \times 65$  pixels extracted from 1,417 testis histology images. There were approximately 200 tubules per histology image. Each tubule image was centered by the label point and cropped to a window size of  $50 \times 50$  from the 2.5x histology image. Then, the tubule images were resized to  $65 \times 65$  for better resolution. Generally, a window size of  $50 \times 50$  in the 2.5x histology image will include the whole tubule. If we set the size too small, the information of the boundary of the tubule will be cropped; on the other hand, the window might include too much information not belonging to a tubule. The details of testis histology images are explicated in section 3.3.1.

To correct for exposure differences, all raw rgb images were normalized by scaling each color channel independently between -1 and 1. In our experiments, the training datasets for pixel labeling and image classification model were independent; most patches used for pixel labeling layer were not drawn from tubules that were used for image classification.

The tubule-classification dataset included five tubule categories: normal, those with few/many vacuoles, those with abnormal germ cells, those with varied degrees of germ cell loss (loss of Elongated Spermatids (ES), loss of ES and Round Spermatids (RS), loss of ES, RS, and earlier, and Sertoli Cell only), and those with germ cells in lumen. Each tubule in our training set was manually

- Normal (no vacuoles, no germ cell loss, no germ cell in lumen)
- Vacuoles: (1) no vacuoles (2) few vacuoles (3) many vacuoles
- Germ cell: (1) normal (2) abnormal
- Germ cell loss: (1) no germ cell loss (2) loss of Elongated Spermatids (ES) (3) loss of ES and Round Spermatids (RS) (4) loss of ES, RS, and earlier (5) Sertoli Cell only
- Lumen: (1) normal (2) germ cell in lumen

(1) Normal / many vacuoles		(3) Normal / Sertoli cell only		(5) Normal / germ cell in lumen	
0	Normal	0	Normal	0	Normal
1	Many vacuoles Normal/abnormal germ cell All degrees of germ cell loss With/without germ cell in lumen	1	No/few/many vacuoles Normal/abnormal germ cell Sertoli cell only With/without germ cell in lumen	1	No/few/many vacuoles Normal/abnormal germ cell All degrees of germ cell loss Germ cell in lumen
(2) Normal / few / many vacuoles		(4) Five stages of germ cell loss			
0	Normal	0	Normal		
1	Few vacuoles Normal/abnormal germ cell All degrees of germ cell loss With/without germ cell in lumen	1	No/few/many vacuoles Normal/abnormal germ cell Loss of ES With/without germ cell in lumen		
2	Many vacuoles Normal/abnormal germ cell All degrees of germ cell loss With/without germ cell in lumen	2	No/few/many vacuoles Normal/abnormal germ cell Loss of ES and RS With/without germ cell in lumen		
		3	No/few/many vacuoles Normal/abnormal germ cell Loss of ES, RS, and earlier With/without germ cell in lumen		
		4	No/few/many vacuoles Normal/abnormal germ cell Sertoli cell only With/without germ cell in lumen		

Figure 4.5: The tubule-classification dataset includes five tubule categories: normal, vacuoles, abnormal germ cell, germ cell loss, as well as germ cell in lumen. Different text background colors represent varied degree of individual category. The dataset of each experiment contains different combinations of five categories. Five experiments include (1) normal / many vacuoles (2) normal / few / many vacuoles (3) normal / Sertoli cell only (4) five stages of germ cell loss (5) normal / germ cell in lumen. The table lists the combinations of five categories for each class defined in five experiments.

Table 4.2: Tubule count distribution over all categories.

Class	Count
Unclassified	250,655
No germ cell loss	37,134
Loss of ES	689
Loss of ES and RS	147
Loss of ES, RS and earlier germ cells (spermatocytes and/or spermatogonia)	227
Sertoli cell only	274
No germ cell loss, Abnormal germ cells	1,004
Loss of ES, Abnormal germ cells	298
Loss of ES and RS, Abnormal germ cells	104
Loss of ES, RS and earlier germ cells (spermatocytes and/or spermatogonia), Abnormal germ cells	69
No germ cell loss, Few vacuoles	1,364
Loss of ES, Few vacuoles	394
Loss of ES and RS, Few vacuoles	152
Loss of ES, RS and earlier germ cells (spermatocytes and/or spermatogonia), Few vacuoles	605
Sertoli cell only, Few vacuoles	455
No germ cell loss, Abnormal germ cells, Few vacuoles	107
Loss of ES, Abnormal germ cells, Few vacuoles	123
Loss of ES and RS, Abnormal germ cells, Few vacuoles	48
Loss of ES, RS and earlier germ cells (spermatocytes and/or spermatogonia), Abnormal germ cells, Few vacuoles	88
No germ cell loss, Many vacuoles	18
Loss of ES, Many vacuoles	17
Loss of ES and RS, Many vacuoles	29
Loss of ES, RS and earlier germ cells (spermatocytes and/or spermatogonia), Many vacuoles	1,240
Sertoli cell only, Many vacuoles	1271
Loss of ES, Abnormal germ cells, Many vacuoles	3
Loss of ES and RS, Abnormal germ cells, Many vacuoles	10
Loss of ES, RS and earlier germ cells (spermatocytes and/or spermatogonia), Abnormal germ cells, Many vacuoles	92
No germ cell loss, Germ cells in lumen	891
Loss of ES, Germ cells in lumen	45
Loss of ES and RS, Germ cells in lumen	6
Loss of ES, RS and earlier germ cells (spermatocytes and/or spermatogonia), Germ cells in lumen	9
No germ cell loss, Abnormal germ cells, Germ cells in lumen	16
Loss of ES, Abnormal germ cells, Germ cells in lumen	3
Loss of ES and RS, Abnormal germ cells, Germ cells in lumen	5
Loss of ES, RS and earlier germ cells (spermatocytes and/or spermatogonia), Abnormal germ cells, Germ cells in lumen	1
No germ cell loss, Few vacuoles, Germ cells in lumen	20
Loss of ES, Few vacuoles, Germ cells in lumen	11
Loss of ES and RS, Few vacuoles, Germ cells in lumen	7
Loss of ES, RS and earlier germ cells (spermatocytes and/or spermatogonia), Few vacuoles, Germ cells in lumen	13
No germ cell loss, Abnormal germ cells, Few vacuoles, Germ cells in lumen	2
Loss of ES, Abnormal germ cells, Few vacuoles, Germ cells in lumen	5
Loss of ES and RS, Abnormal germ cells, Few vacuoles, Germ cells in lumen	1
Loss of ES, RS and earlier germ cells (spermatocytes and/or spermatogonia), Abnormal germ cells, Few vacuoles, Germ cells in lumen	4
Loss of ES, RS and earlier germ cells (spermatocytes and/or spermatogonia), Many vacuoles, Germ cells in lumen	5
Sertoli cell only, Many vacuoles, Germ cells in lumen	1
Loss of ES, RS and earlier germ cells (spermatocytes and/or spermatogonia), Abnormal germ cells, Many vacuoles, Germ cells in lumen	2
Normal rete testis	905
Abnormal rete testis	205
Normal blood vessel	3,494
Abnormal blood vessel	304
Tumor	2
Cyst	12
PAS+accumulations	191

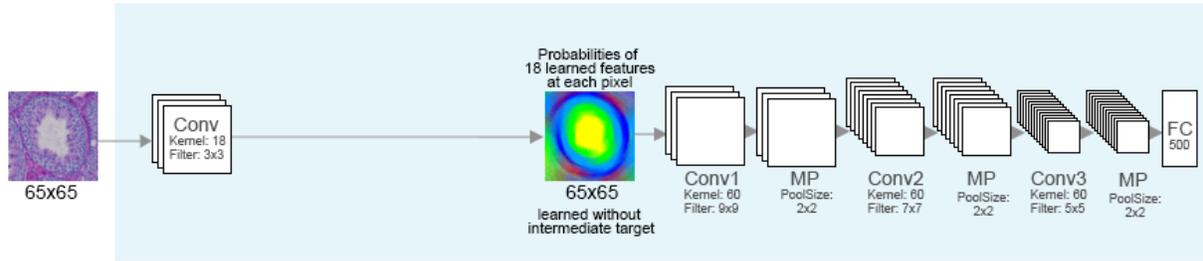


Figure 4.6: CNN without hints: the classification task is trained without intermediate target and directly backpropagate the parameters from top to bottom.

annotated by an expert in male reproduction, indicating different categories and combinations. From 1,417 testis histology images, 302,777 tubule image patches were identified by the pre-segmentation process, where 37,134 patches are labeled normal out of 52,122 labeled patches, more information of tubule counts listed in Table 4.2. I conducted five experiments over abnormal tubules labeled with various types of qualitative and quantitative germ-cell loss, along with normal controls. The dataset settings for five experiments are depicted in Figure 4.5.

I designed five individual datasets for five different experiments, and each was composed of 7,000 images. The effective size of any class can be increased by rotations and reflections, and this was done where needed to achieve a balanced training dataset. Each dataset was partitioned into six equal sized subsamples, with one used for validation, one for testing, and the remaining four for training. The abnormal tubule types were not independent (a single tubule can be labeled as multiple classes).

To evaluate our experimental results, I compared my approach to a classical deep-network VGG-16 (Simonyan and Zisserman, 2014) and a second CNN model with an equivalent topology to our network. For VGG-16, the input images were extracted from the original 20x histology image and resized to  $224 \times 224$  (a requirement of the model). The VGG-16 was initialized with weights from a pre-trained model and the entire network was fine-tuned for our tasks. The compared CNN model is illustrated in Figure 4.6. It was designed to have the same architecture and number of parameters as our two-stage curriculum learning model, but without hints. Its weights were trained

Table 4.3: Pixel labeling results. In the top 3 scores, our pixel labeling model can predict above 90% accuracy.

Label	Top 1	Top 2	Top 3
Background	91.99%	97.09%	99.03%
Tunica	98.23%	99.33%	99.78%
Interstitial Space	67.71%	86.10%	95.29%
Leydig Cells	76.54%	88.39%	93.36%
Vessel	87.98%	96.15%	98.32%
Basal lamina	85.91%	92.15%	96.07%
Sertoli	89.10%	95.50%	96.68%
Spermatogonia	84.62%	88.46%	94.23%
Spermatocytes	79.85%	92.72%	96.36%
Early Spermatids	77.48%	94.19%	97.58%
Late Spermatids	73.70%	89.57%	96.68%
Lumen	83.86%	93.01%	97.11%
Rete	73.99%	91.89%	97.14%
PAS accumulation	95.25%	97.62%	99.52%
Abnormal/Tumor	0%	0%	0%
Other	80.48%	94.46%	97.83%
Cells in Lumen	81.75%	91.73%	95.86%
Vacuole	86.84%	95.93%	98.56%

using classical back-propagation. It directly predicted tubules classes using the same training images as our model.

#### 4.4 Experiment Results

This section presents the results of pixel labeling and tubule classification. The results demonstrate that the pixel labeling model outperforms traditional CNN. Also, the sensitivity analysis on pixel labeling accuracy to image classification performance is presented in this section. The results show that the image classification model is very robust and does not require extremely accurate pixel labeling.

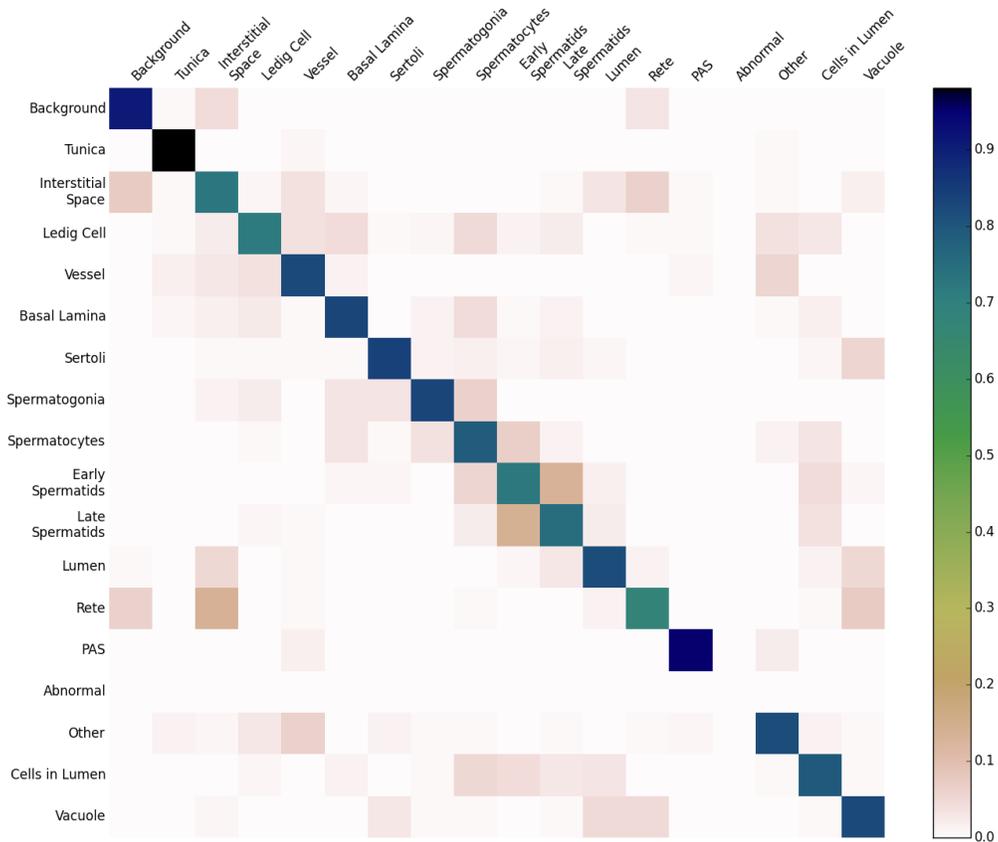


Figure 4.7: Confusion matrix of pixel labeling. Rows are the ground truth, and the columns are the predicted results.

#### 4.4.1 Pixel Labeling Results

The results of our pixel-labeling are reported in Table 4.3 according to the rank of the training label among the estimated label likelihoods. For the top 1 score, the accuracy varies from 67% to 98%. But, when the top 3 of 18 predictions are considered all labels are above 90% of accuracy. The low accuracy of the *other* category reflects a lack of consistent training examples. Since background, rete testis, lumen, vacuoles, and interstitial space patch classes all characterize different types of empty space, they are often confused, leading to reduced accuracy. In fact, they can even confuse experts without sufficient context. However, in cases where they are distinguishable, they provide valuable information for localizing the target pixel. Figure 4.7 illustrates the confusion matrix between assigned and predicted labels. Rows are the ground truth while columns are the prediction results. Here the relative confusion between similar label types (background, rete testis, lumen,

Table 4.4: Sensitivity analysis of pixel labeling classifier on five categories, where rows are the corrupted rate on pixel labeling results.

	0%	10%	20%	30%
(1) Normal / many vacuoles	99.138%	98.621%	98.621%	98.534%
(2) Normal / few / many vacuoles	75.862%	72.845%	73.103%	73.103%
(3) Normal / Sertoli cell only	99.138%	98.103%	98.362%	98.103%
(4) Five stages of germ cell loss	58.103%	54.310%	53.448%	53.276%
(5) Normal / germ cell in lumen	96.121%	96.121%	96.121%	96.121%

vacuoles, and interstitial space) is evident. Physically adjacent label classes also tend to be confused as suggested by the relatively high probabilities of confusion along super and sub diagonals.

Note that the pixel labeling is not directly used by the classifier. Instead, a vector of pixel-label probabilities are provided as hints for downstream image classification. I hypothesize, and have some evidence based on limited testing, that image classification does not require extremely accurate pixel labeling as discussed in Section 4.4.2.

Visualizations of the pixel-label probabilities within an tubule image are shown in Figure 4.8. The images in Figure 4.8(b)(c) show the probability maps of particular labels, and (d) illustrates the label with the highest probability at each pixel. After labeling, it becomes evident that the testis separates into many different classes, which helps answer higher level problems such as which are normal and which are abnormal.

#### 4.4.2 Sensitivity Analysis of Pixel Labeling Classifier

I hypothesized that image classification does not require extremely accurate on pixel labeling. The pixel labeling layer is rather regarded as a vector of probabilities that provide hints for image classification. To test this hypothesis, I conducted experiments on five tubule categories with various corruption rates as shown in Table 4.4. I corrupted the input by assigning neutral pixel label probabilities at random locations and fed the corrupted results to the classification model. From Table 4.4, although accuracy decreases as the corruption rate increases, classification accuracy is only slightly reduced. The pixel labeling model not only consider all label probabilities at the center

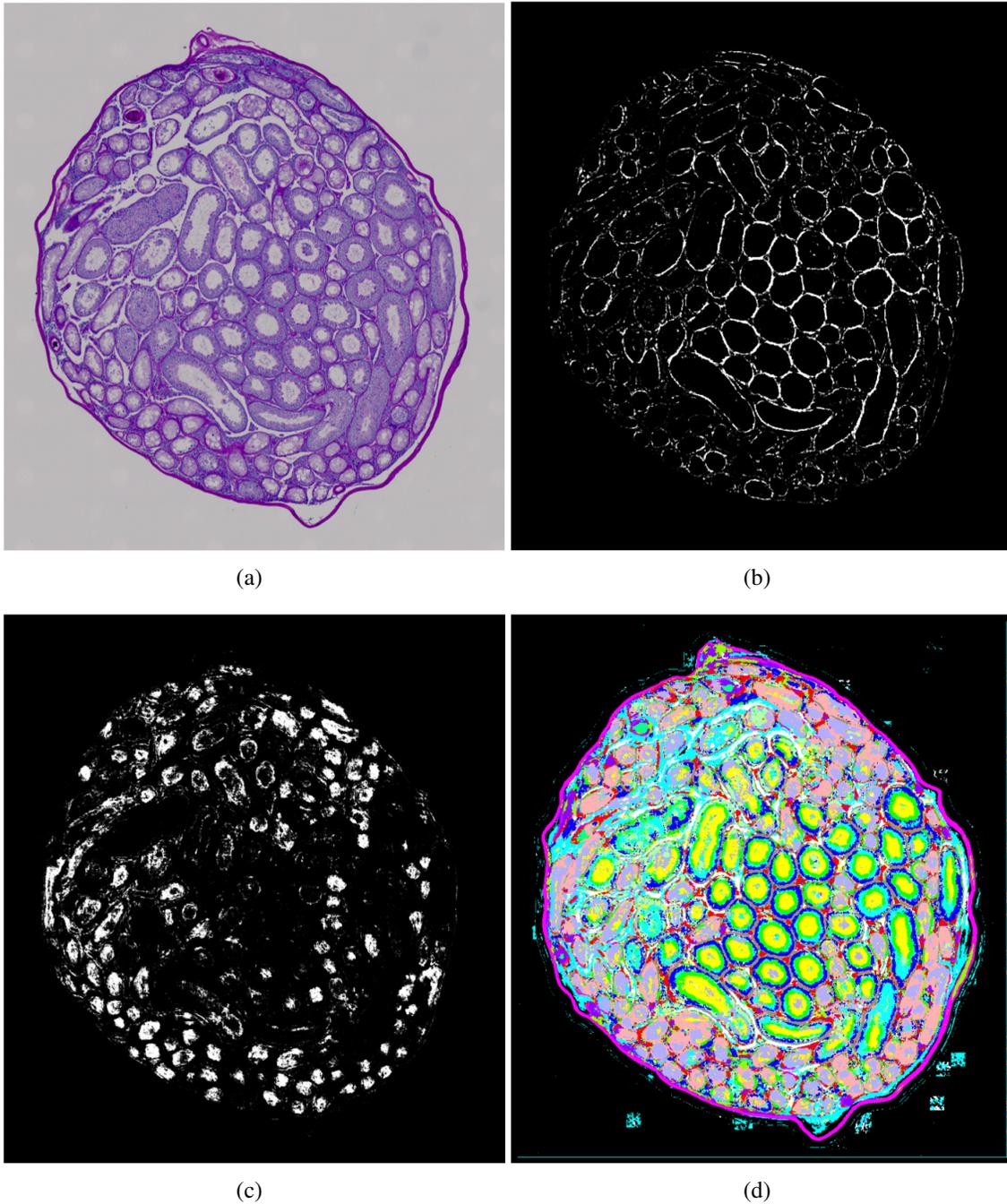


Figure 4.8: Visualizations of the learned pixel labeling model. (a) Example of a true-color testis histology cross section. (b)(c) The probability map for a single pixel-label class (basal lamina and vacuole). Brighter colors indicate higher probabilities. (d) is a false-color image indicating the label with the highest probability at each pixel where pink represents vacuoles, yellow indicates lumen and magenta is tunica. Pink, yellow and magenta clearly highlight regions where the pixel labeling is accurate. Concentric rings of labels within tubules are also mostly as expected.

Table 4.5: Comparison of classification performance on six datasets. I compared my method with VGG-16 and CNN without hints. The best results are in bold typeface.

	Random	VGG-16	CNN	Ours
Normal / many vacuoles	50%	68.845%	89.444%	<b>99.138%</b>
Normal / few / many vacuoles	33%	38.879%	62.688%	<b>75.862%</b>
Normal / Sertoli cell only	50%	60.862%	86.787%	<b>99.138%</b>
Five stages of germ cell loss	20%	21.983%	44.699%	<b>58.103%</b>
Normal / germ cell in lumen	50%	93.966%	90.751%	<b>96.121%</b>

pixel but also take all label probabilities in a neighborhood into consideration when training the classification. In addition, the pixel labeling layer does not give hard labels only to the classification model, instead, a probability distribution of all labels, so that even if the labeling results have lower accuracy, they still provide valuable information to the subsequent task.

#### 4.4.3 Classification Results

For medical images, a tissue contains multiple cell types which may be associated to form specific structures, such as blood vessels or seminiferous tubules. Classifying a tissue involves an overall assessment of cells contained in the tissue. Therefore, if a classifier that can capture the features of cell types in the tissue, the classification performance is apt to improve significantly. Table 4.5 shows the classification performance in terms of accuracy on six sub-datasets. In the first, third, and fifth experiments, the features of normal tubules and the last stage of the corresponding category are more distinct, thus the classification performs better. In the second and fourth experiments which involve quantitative categories, the ambiguities among stages cause the performance to drop off. Nonetheless, our approach outperforms classical methods without hints by more than 10% of accuracy. This suggests that predicting appropriate intermediate feature targets makes learning easier and generalizes to other tasks. I also hypothesize that appropriate intermediate features are not trivially inferred by classical deep-learning architectures as illustrated by our CNN model. Moreover, deep networks like VGG-16 have similar problems. They tend to learn many features unrelated to the final task when leads to poor performance in our experiments.

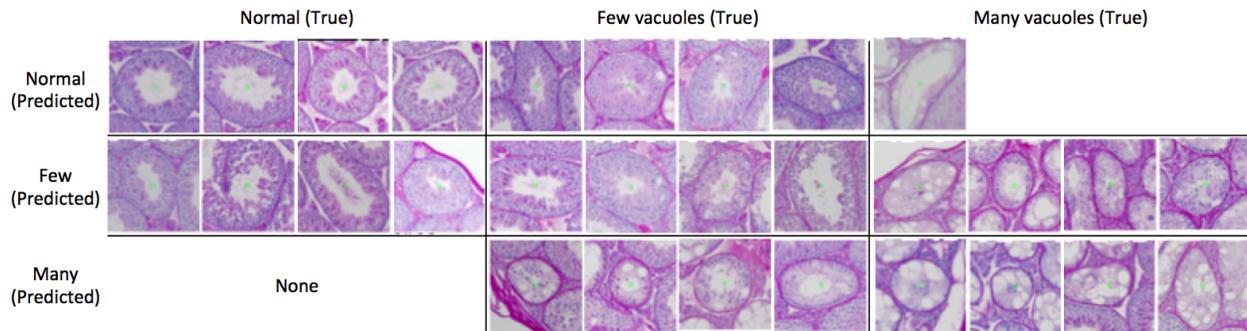


Figure 4.9: Examples of our classification results compared to the ground truth. The examples in first two columns are normal tubules, followed by tubules with few vacuoles, and tubules with many vacuoles. The normal tubules that the model identified with vacuoles are pardonable since most appear to exhibit vacuole-like features. The tubules with vacuoles that the model classified as normal tended to have few vacuoles, thus, leading to ambiguity.

Figure 4.10 shows the image classification results of five experiments. In particular, Figure 4.9 illustrates the examples of classification results in the second experiment. The normal tubules the model identifies as tubules with few/many vacuoles, or the other way around, are extremely hard to classify. Since there are no clear thresholds in terms of the size and the number of the vacuoles in the tubules defined for these classes, it is very challenging to distinguish boundary cases. In the fourth experiment, which distinguishes degrees of germ cell loss, it is difficult to distinguish the boundary cases between adjacent progressive states. For instance, the examples of class 1 is easy to get confused with the ones of class 0 and 2, and class 3 with class 2 and 4, etc.

## 4.5 Conclusion and Discussion

In this chapter, I have presented a practical hidden layer, pixel labeling layer, for curriculum learning. I demonstrate that the pixel labeling model can provide robust and meaningful representations to the subsequent task and challenging task, such as classifying medical images, can be learned efficiently. Also, I demonstrate that our network with hints outperforms other state-of-the-art approaches on the medical image dataset. To evaluate whether the pixel labeling generalizes, I applied the pixel labeling model to different vision problems, which is presented in Chapter 5.

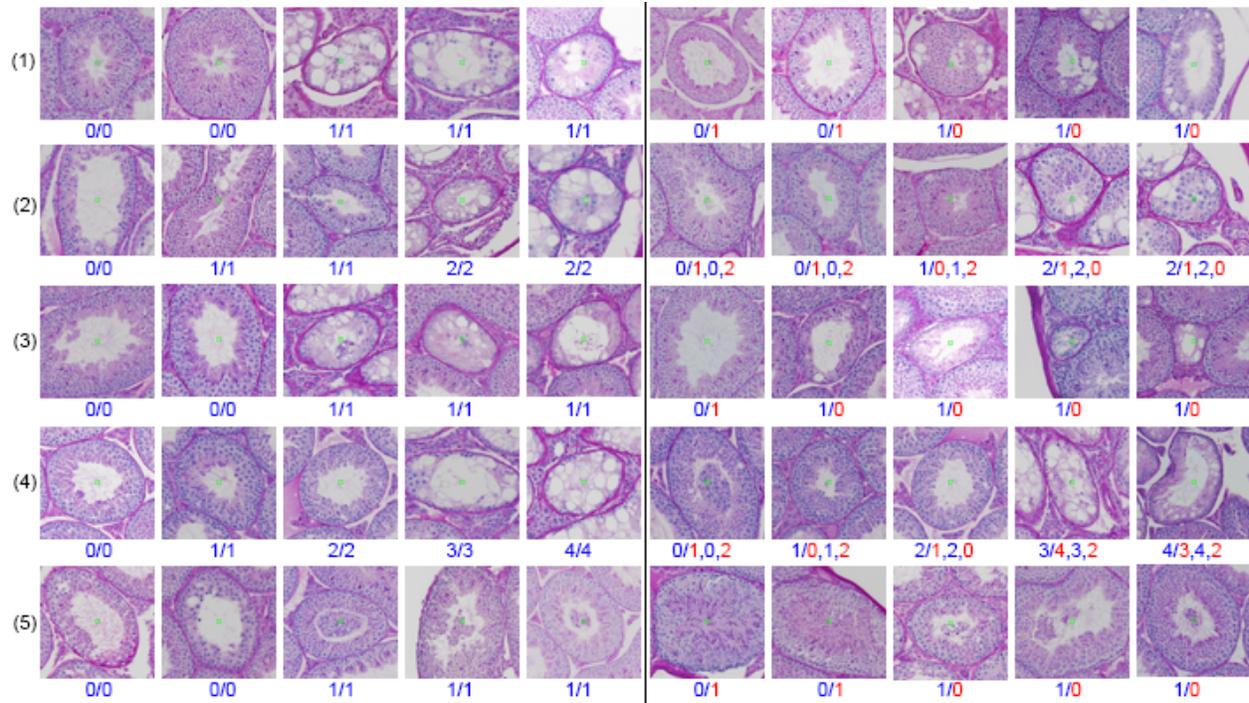


Figure 4.10: Image classification results of five experiments. The results of five experiments are arranged in five rows. The first five columns show the examples of true positive and true negative, while the false positive/negative examples are illustrated in the last five columns. The text below image shows the class of the truth versus prediction, where blue means correct prediction while red is wrong. The numbers are different classes within each experiment defined in Figure 4.5. In the second and fourth experiments, they include degrees of abnormalities, leading to the difficulty to distinguish the boundary cases among states. For instance, image in row four and column seven has features similar to the confused class 0, and image in column eight and class 1 are very much alike.

Still, many challenges remain. Currently, the selection of meaningful labels and the dataset for pixel labeling layer are manually defined by the experts. In addition, I plan to learn the *labels* by using statistical information of the data, and let the data itself suggest the labels for the intermediate layer, which is introduced in chapter 6. Lastly, more work is needed to develop a better solution of generating a suitable dataset for pixel labeling layer automatically.

## CHAPTER 5: GENERALIZING PIXEL LABELING APPROACH

### 5.1 Introduction

Inspired by curriculum learning (Gülçehre and Bengio, 2016), I introduced in Chapter 4 an intermediate learning layer, which learns a supervised *pixel labeling* to provide hints to subsequent image classification layers. I demonstrated that pixel labeling model can provide useful and meaningful hints to the subsequent tasks, and I also showed that the model with the hints learned from pixel labeling layer outperforms other approaches.

In this chapter, I address the question of whether the pixel labeling methods I have developed generalize to other problems? To test if my model generalizes, I introduce two toy problems. One is the Pentomino problem, which is adapted from the original curriculum learning paper (Gülçehre and Bengio, 2016), that tries to classify whether the types of three Pentominoes in an image are the same or not. I added a pixel labeling layer to their model architecture and used their Pentomino dataset for comparison. The pixel-labeling approach predicts a probability vector for every pixel within an aperture window and provides the predicted probabilities to the following tasks as the intermediate hints. The goal of the toy problem is to evaluate whether the pixel labeling layer can accomplish a similar improvement in the Pentomino learning task when compared to the hint layers used in the original paper. The results show that pixel labeling improves the Pentomino classification performance either with or without their proposed hints.

However, since the Pentomino problem uses binary images as shown in Figure 5.1(a), the Pentomino dataset misses an important characteristic that I believe is crucial to the pixel-labeling used in the histology image dataset, which is that the state of a biological organism is reflected by local variability in pixel intensities and textures. In addition, Pentomino types are simple, monochrome, and many local patterns are shared among different Pentomino types. This makes the

Pentomino problem a less ideal model for local context. Therefore, I designed a second toy problem that is equally hard to learn and can benefit from labeling hints. The new dataset is generated by randomly selecting two textures from a well known texture dataset, Brodatz Album (Brodatz, 1966), representing foreground and background textures in an image. The leaving objective of this second toy problem is whether the foreground and background textures in an image are the same or not. The goal of this toy problem is to validate whether the labeling model can generalize to datasets where texture carries significant information and can make significant improvements in terms of classification performance.

## 5.2 Pentomino Dataset

In order to evaluate the generalization of the pixel labeling model, I first demonstrated it on a toy Pentomino dataset for object recognition using  $64 \times 64$  binary images. I chose the Pentomino dataset because it was used by the Gülçehre and Bengio (2016) in a setting of curriculum learning. Also, it is simple and can generate countless images for use as ground truth that I can easily apply to our model. The implementation of dataset generation is based on the source code provided by Gülçehre and Bengio (2016).

### 5.2.1 Problem Definition

Figure 5.1(a) illustrates the ten types of Pentominoes used in the dataset. The foreground pixels are labeled as 1 (white), and background pixels as 0 (black). Each task image contains three Pentominoes and the ultimate goal of this task is to classify if all the Pentominoes in an image are of the same type or not. To avoid the dataset from being biased, the number of samples with the same or different Pentomino types were equal. Three of 16 different block positions are selected at random, without replacement, and Pentominoes are each placed into the chosen blocks. Figure 5.1(b) is an example of an image where all Pentominoes are of the same type, while Figure 5.1(c) includes different Pentomino types.

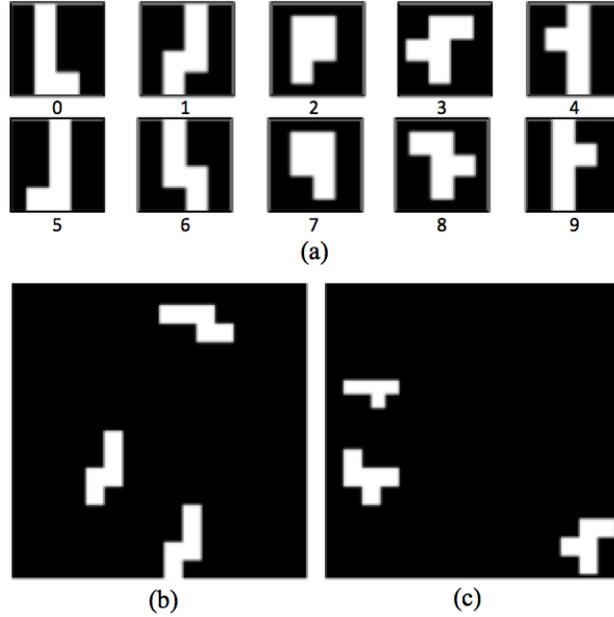


Figure 5.1: (a) Different types of Pentominoes used in the dataset (Gülçehre and Bengio, 2016). Example images from our modified dataset showing (b) same Pentomino type and (c) different Pentomino types.

## 5.2.2 Experiment Setup

In this section I describe how to apply a pixel labeling layer to the original model architecture designed in Gülçehre and Bengio (2016). Also, I will show the configuration details for generating the Pentomino dataset for the experiments.

### 5.2.2.1 Block Size Selection

In their original paper, Gülçehre and Bengio (2016) divided a  $64 \times 64$  uniform grid image into  $8 \times 8$  blocks of  $8 \times 8$  pixels. Unlike their setup, the size of a block in my dataset is twice as big as theirs, leading to a total of 16 blocks ( $4 \times 4$ ) with block size of  $16 \times 16$  pixels. In both cases, the Pentomino is centered in the block without translation. A Pentomino is guaranteed to be entirely contained within a block, so there are no overlaps between Pentominoes.

In the pixel labeling model, I chose to employ a small aperture window of size  $5 \times 5$  pixels. Figure 5.2 illustrates examples of aperture window on the sample images in red squares. The pixel labeling model predicts the possible Pentomino type on the center pixel of the aperture window

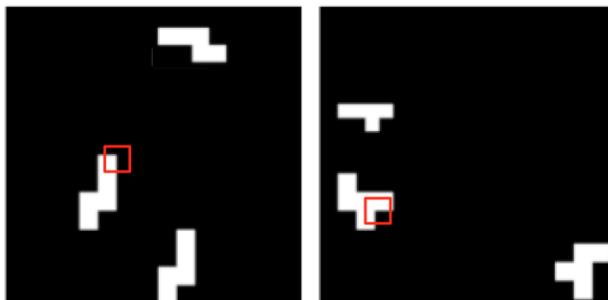


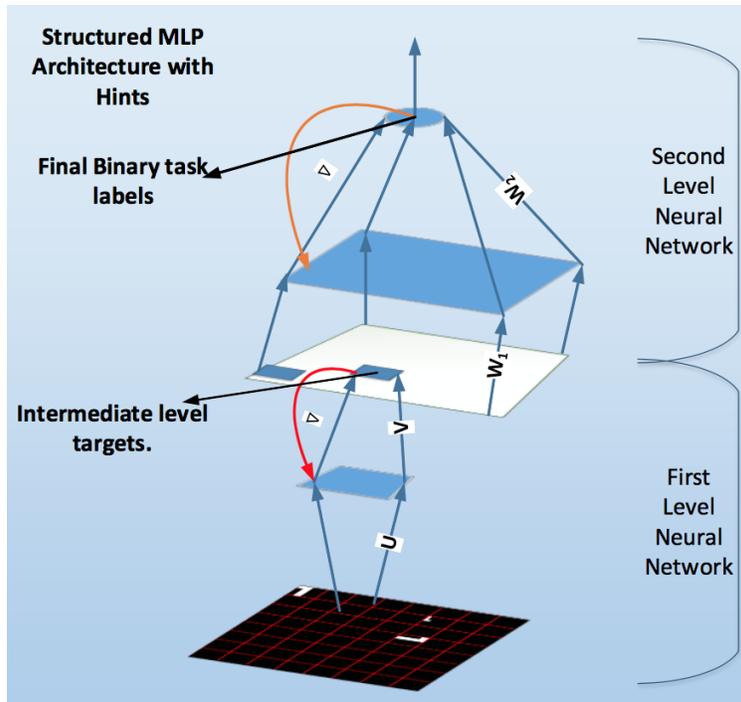
Figure 5.2: Red square in the images are example of aperture windows using in the pixel labeling model. The pixel labeling model predicts the possible Pentomino type on the center pixel of the aperture window in the context of its surrounding neighbors.

given its surrounding neighbors. I centered an aperture window on each pixel in the image. Each aperture window is labeled after considering all the pixels in it. The resulting label is associated with the center pixel of the aperture window. These labels are used as hints for the subsequent learning task.

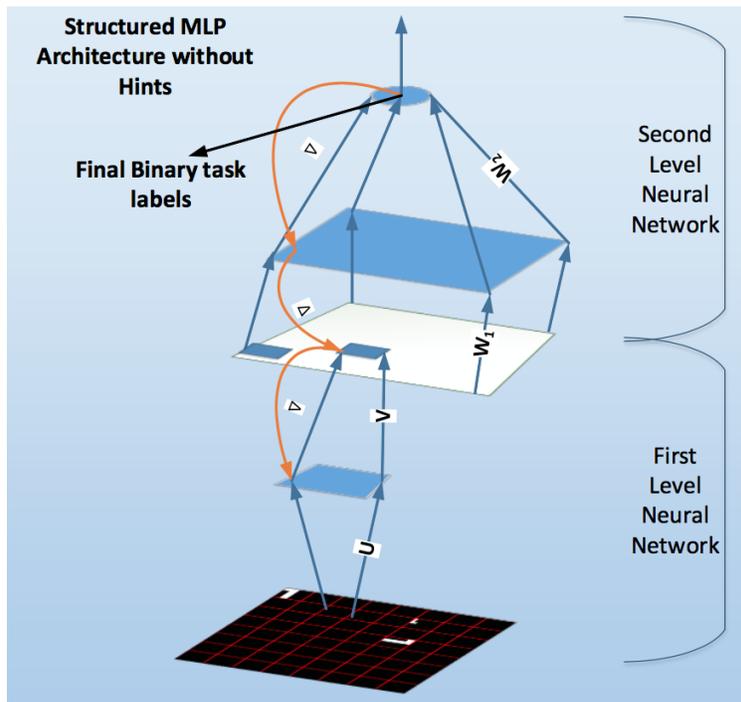
Each Pentomino has two different types of transformations, including rotation by four different angles and scaling with two factors, yielding eight different possible transformations. In my experiments, the scaling factor is changed to  $\{3, 4\}$  to ensure the aperture used for the pixel labeling model does not cover a whole Pentomino. This increase in Pentomino and block size was done so that the pixel-level labeling task would only be able to see part of any Pentomino within its aperture.

### 5.2.2.2 Original Model Architecture

Gülçehre and Bengio (2016) proposed a network architecture with a different learning process to evaluate the value of an intermediate *hint* layer. The architecture of the network is separated into two parts as illustrated in Figure 5.3. The first part, Part 1 Neural Network (P1NN), has shared weights and local connectivity. The input of P1NN is the patch of the image, and the output of it is a vector with 11 labels (10 types of Pentomino and empty) per patch. The expected output of P1NN is a 11-label output vector per block indicating the Pentomino class for every block. The task for P1NN is to recognize the Pentomino type, and locate each Pentomino block in the image.



(a)



(b)

Figure 5.3: (a) Structured MLP architecture with hints (SMLP-Hints) (Gülçehre and Bengio, 2016). P1NN are bottom two layers and top two layers are P2NN. P1NN and P2NN are trained separately, not jointly. In SMLP-Hints, P1NN is trained on each  $8 \times 8$  patch extracted from the image and the softmax output probabilities of all 64 patches are concatenated into a  $64 \times 11$  vector that forms the input of P2NN. (b) Structured MLP architecture without hints (SMLP-Nohints) (Gülçehre and Bengio, 2016). It is the same architecture as SMLP-Hints but with P1NN and P2NN trained jointly in terms of the final classification task.

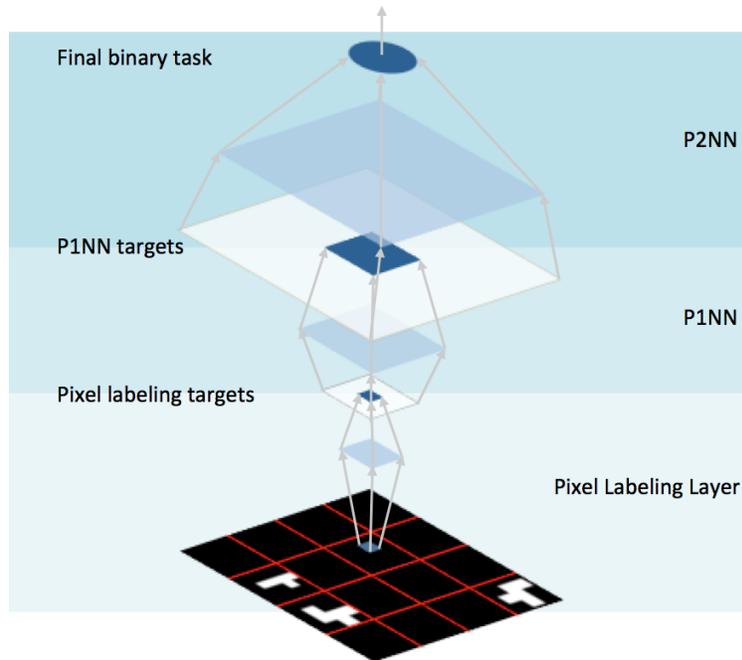


Figure 5.4: SMLP architecture using pixel labeling hints. Bottom two layers are pixel labeling layers, while middle two and top two layers are P1NN and P2NN, respectively. Pixel labeling layer is trained on  $5 \times 5$  patch extracted from the image and outputs probabilities for each of the 11 possible Pentomino at every pixel in the image.

The second part, P2NN, is a fully connected layer, taking the concatenated output of P1NN over all patches as the input, and determining whether all Pentominoes in the image are the same or not.

To test the hypothesis, they conducted two experiments using this architecture with two different training approaches. The main difference between the two approaches was whether P1NN and P2NN were trained separately or jointly. The first one is called SMLP-Hints as shown in Figure 5.3(a), which exploits a hint about the types of Pentomino, indicating a semantics for the outputs of P1NN. P1NN is trained with the intermediate target; therefore, P1NN and P2NN in this model are trained separately. The other one is SMLP-Nohints as shown in Figure 5.3(b), where P1NN and P2NN are trained jointly and the model does not use the intermediate target to specify the types of Pentomino.

### 5.2.2.3 Pixel Labeling Layer

Based on the two models (with hints/without hints), I replaced the Pentomino classification layers with pixel labeling layers below P1NN and came up with two similar models. In my experiments, the architecture consists of three subnetworks as shown in Figure 5.4. The first two layers are the pixel labeling layer predicting the probability vector for every pixel within an aperture window, where the probability vector contains the probabilities of 11 possible Pentomino labels. Pixel labeling layer is trained on  $5 \times 5$  patch extracted from the image and outputs 11-label probabilities vector for the center pixel of the patch. The output of the pixel labeling layer then forms the input block of P1NN of size  $16 \times 16 \times 11$ . In layers three to six, the architectures are identical to P1NN and P2NN in the original architecture. In summary, the pixel labeling layer is trained to learn the hints of possible Pentomino types at the pixel level. P1NN learns to specify the type of Pentomino present at 16 blocks of the image. Finally, P2NN predicts the results indicating whether three Pentominoes in an image are of the same or different types.

The pixel labeling layer is a standard CNN with one convolutional layer (11 kernels with filter size of  $3 \times 3$ ), followed by two fully-connected layers. To make sure that the input image patch for the pixel labeling layer does not cover the whole Pentomino, I chose a patch size of 5 for the experiment. Hyperparameter values were optimized by random search. I randomly selected the number of hidden units from [50, 100, 200, 300, 500, 1024, 2048] and randomly selected the learning rates within the interval of [0.001, 0.5]. Two fully-connected hidden layers with 1024 and 300 hidden units are chosen for the experiments. The dataset composes 20,000 patches in total, partitioned into six equal sized subsamples, including validation and testing dataset each using a partition, and training data using the remaining four.

In the pixel labeling, the model classifies the center pixel given a patch window of context. Since the shapes of Pentomino are monodrome, the pattern of one patch can be part of many different Pentomino types. Therefore, for a given image patch, I formed a target vector reflecting the frequency of particular patch pattern appearing in 11 labels (10 types of Pentomino and empty). To obtain the frequency of patch pattern appearing in 11 labels, I used a sliding window of size 5

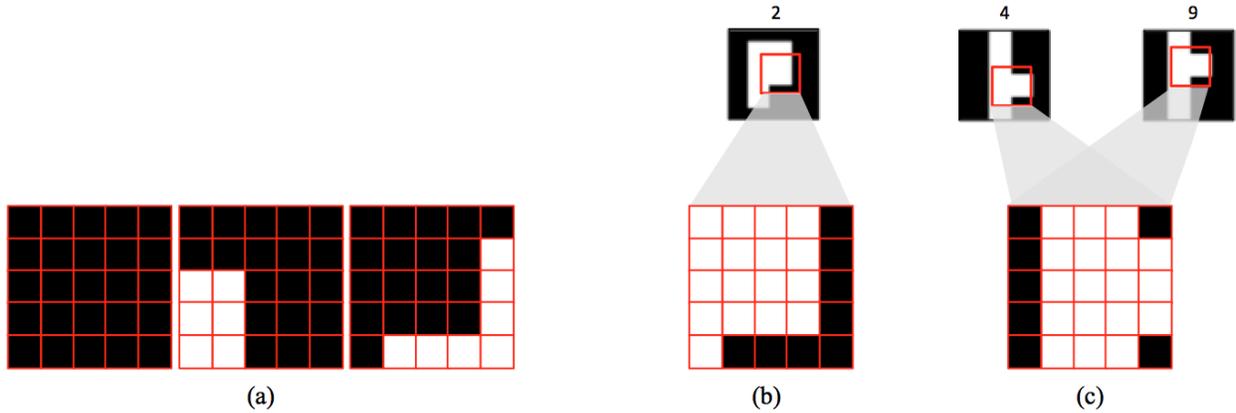


Figure 5.5: (a) Examples of background patch pattern. (b) An example of patch pattern that is unique to Pentomino 2. (c) An example of patch pattern that is not unique to any one Pentomino. Both Pentomino 4 and 9 could have such pattern, leading to ambiguity while labeling the pixels.

traversing every pixel in the Pentomino block. If such patch pattern shows five times in Pentomino 1 and ten times in all Pentominoes, for example, the value of Pentomino 1 in the target vector for this particular patch will be 0.5.

Figure 5.5 illustrates some examples of the patch patterns. Figure 5.5(a) shows the pattern examples of the background. It is noteworthy that the background pattern is not always all black. If the center pixel of the patch pattern is not within a Pentomino, the patch is identified as the background, because the target the model tries to classify is the center pixel of the patch, and the surrounding pixels are only provided as the context as shown in Figure 5.5(a). Figure 5.5(b) illustrates a patch pattern that is unique to Pentomino 2. Thus, the target vector for this patch will be all 0 except 1.0 for Pentomino 2. There are also many patterns like Figure 5.5(c) that appear in many Pentominoes. An exhaustive list of all possible patch patterns that appear in the ten Pentominoes, includes only 284 distinct patterns in a  $5 \times 5$  window size. Of these 24 are unique to a single Pentomino, and 76 are common to every Pentomino. This suggests that most of the patterns are shared among many Pentominoes, making our labeling task harder and the task more challenging.

Lastly, the goal for the pixel labeling layer is to provide hints for PINN to identify the type of the Pentomino. As a result, the model learns the parameters by minimizing the difference between

the predicted and target frequency distributions and output a probability distribution over 11 labels for each pixel.

### 5.2.3 Experiment Results

In this section, I show the improvements to the P1NN and P2NN results after applying pixel labeling layer to the model, where the binary image of the P1NN input is replaced by a probability vector for each pixel. In addition, the sensitivity analysis of the pixel labeling layer is also presented. The results prove that the pixel labeling is a practical and robust source of hints.

#### 5.2.3.1 Pixel Labeling Hints

Recall that the Pentomino problem infers a binary classification task, where Gülçehre and Bengio (2016) tested several machine learning algorithms on the Pentomino dataset, most had very poor performance, generally no better than the results of a coin-flip. The best results of the model learned without hints can only reach 64% accuracy in a dataset with 40,000 test examples, but the accuracy significantly increases to around 97% when layers 1-3 were pretrained using hints.

Before setting off to learn a pixel labeling model, I first examined how the original model would perform if it was provided the exact Pentomino probability at every pixel. Would this information be beneficial to P1NN for identifying the Pentomino for the block, or, even more, is it useful to P2NN for classifying whether the Pentominoes in the image were the same type or not? To validate this hypothesis, I conducted a simple experiment. As described in 5.2.2.3, if I set a small aperture window of size  $5 \times 5$  traversing all 10 Pentominoes, I obtained 284 distinct patterns. Recall the images are binary, and the 284 are a subset of the  $2^{25}$  possible patterns. The target vector is obtained by examining whether the particular patch pattern appears in a given Pentomino type. For example, if a patch pattern appears in Pentomino 2 and 3, the target vector for such patch pattern will be 0.5 on Pentomino 2 and 3. I calculated the probabilities for each 11 Pentomino labels for each individual pattern and assigned the likely Pentomino probabilities to every pixel in P1NN input. The accuracy of the identification task in P1NN was increased to 98.182%, and the performance for

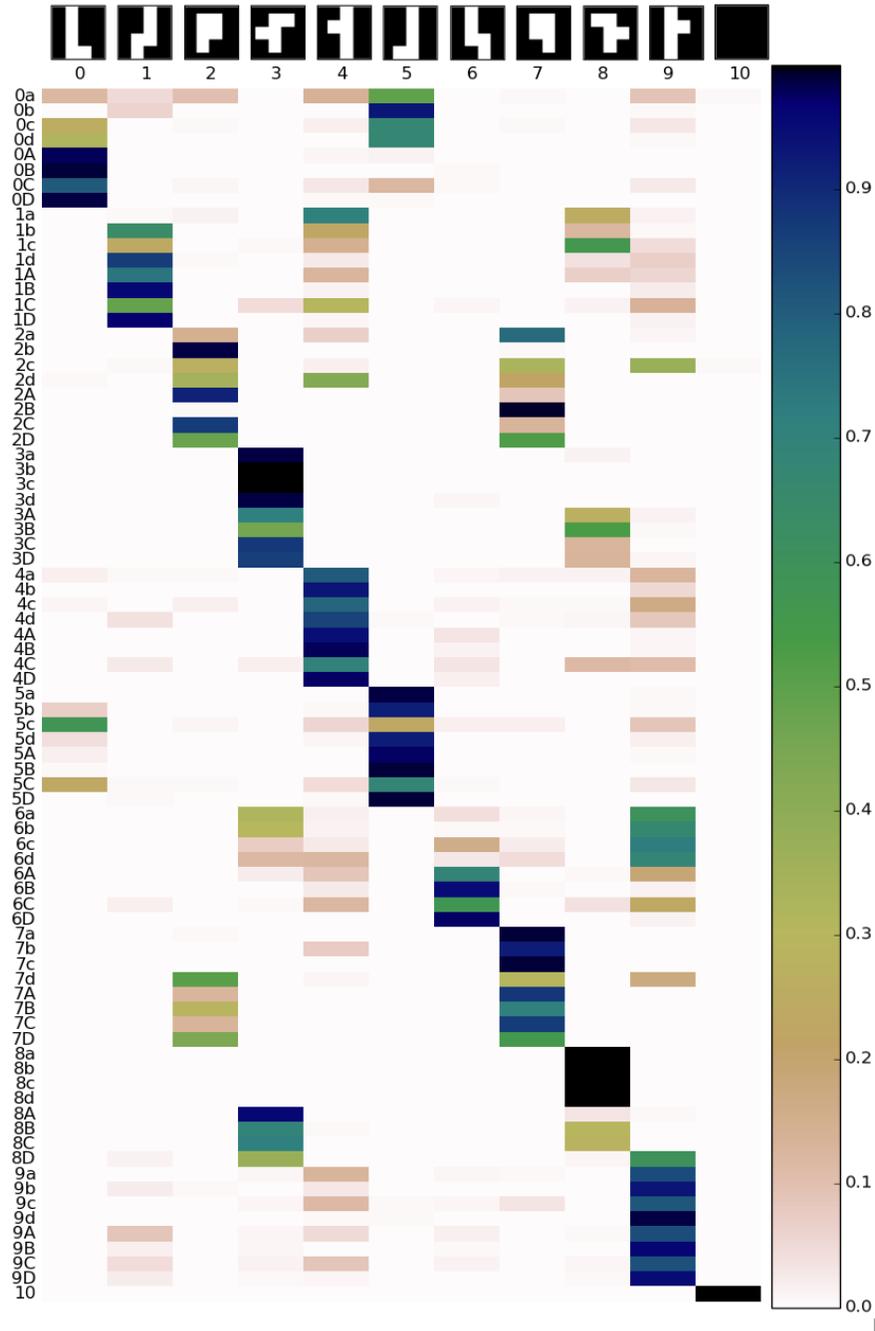


Figure 5.6: Confusion matrix of the pixel labeling on all transformations of Pentominoes. Columns are the predicted results of the pixel labeling model, and rows are the ground truth of ten Pentominoes with eight transformations plus background (10), where the numbers represent ten Pentomino type (0-9) and the letters indicate the eight transformations (a-d, A-D). Lowercase (a-d) are small Pentominoes (scale 3) rotated by four angles and uppercase (A-D) are four rotated large Pentominoes (scale 4). The values in the confusion matrix are the normalized predicted probabilities of each transform Pentomino being particular Pentomino type, where white means low probability and black illustrates high probability.

Table 5.1: Image classification results of two SMLP learning processes in 20k dataset compared with applying pixel hints or not.

	Without Pixel Hints	With Pixel Hints
SMLP-NoHints	53.836%	73.02%
SMLP-Hints	67.323%	73.44%

final P2NN task increased to 91.516% accuracy. This simple experiment demonstrates that pixel labeling can be used as prior knowledge, guiding the model learning in the right direction.

I then attempted to learn a pixel labeling model with a standard CNN architecture, and applied it to P1NN on the modified Pentomino dataset. Using the original SMLP architecture, I got 79.533% accuracy on P1NN intermediate task, block Pentomino identification, in 20k dataset. After applying the pixel labeling to SMLP, the performance improves to 87.189%.

Figure 5.6 shows the confusion matrix of the pixel labeling results on P1NN task. The rows in the confusion matrix are the normalized predicted probabilities of each Pentomino type given a Pentomino orientation and scale, where white means low probability and black illustrates high probability. Half of the Pentomino shapes are the mirror images of the other half. For example, Pentomino 5 is the mirror image of Pentomino 0, Pentomino 1 and 6 are mirror-image pair, and etc. Because the feature representations of those Pentomino mirror-image pairs are similar, this often lead to confusion between mirror versions. The transformed Pentomino are correctly identified in the vast majority of examples, achieving 72.84% accuracy. It is obvious that, with the pixel labeling hints, P1NN can easily classify the type of Pentomino, even the Pentomino is in different transformations.

Table 5.1 shows the results of SMLP-Nohints and SMLP-Hints with and without adding the pixel labeling hints on the meta task, whether the types of Pentominoes in the image are the same or not, in 20k dataset. When the model learns without any hints, the learning procedure appears to search the parameter space blindly not knowing what to look for, and seems to fall into local minimum that leads to an accuracy indistinguishable from random guessing. However, with a simple hint like the one proposed in (Gülçehre and Bengio, 2016), the meta-task accuracy improves.

Table 5.2: Sensitivity analysis of pixel labeling hints and P1NN hints on image classification task. Columns represent the corruption rates applied to the pixel labeling results.

	0%	10%	20%
Corrupted pixel labeling hints	73.44%	72.56%	70.20%
Corrupted P1NN hints	73.44%	73.44%	71.80%

Moreover, applying the pixel hints to SMLP-Nohints architecture increases the performance by an additional 19.184% and by 6.117% in SMLP-Hints architecture. This suggests that if the model can learn with an appropriate guidance, it will learn better and reduce classification errors as well. Although I expected that including the pixel labeling hints in SMLP-Hints architecture would improve the performance even more, the best result I achieved was 73.44%. Because of the ambiguity of Pentomino patterns when viewed through an aperture the hints provided by the pixel labeling have limitations to give P1NN exact answers of which Pentomino type it identifies. But overall, from the experiments on P1NN and P2NN tasks, they both demonstrate that the pixel labeling hints provide beneficial information to all the downstream tasks and improve the performance significantly.

In the original paper, SMLP-Hints had 69.3% accuracy in 20k dataset and 96.9% in 40k dataset. When I compared my results with SMLP-Hints in 20k dataset, the pixel labeling hints improved their Pentomino-discrimination hints by around 5%, but does not outperform their hints in 40k dataset. Although they claim that adding more training examples did not help for both training and test error, my experiments do not support the assumption using SMLP-Hints. Moreover, their model relies on each Pentomino falling within predefined image regions which is hard to achieve for real classification problems without applying a segmentation or object detection task beforehand; in contrast, the pixel labeling hints do not require objects to be segmented in advance. The hints provided by the pixel labeling classifier is therefore more feasible and practical in a general setting.

### 5.2.3.2 Sensitivity Analysis on Pixel Labeling Classifier

My previous results on the histology images suggest that image classification does not require a perfect pixel labeling. To test if this still holds for the Pentomino toy problem, I conducted the experiments with various corruptions applied to the pixel labels and P1NN hints as shown in Table 5.2. I corrupted the results of the pixel labeling by replacing the label vectors with neutral probabilities at the random locations. Similarly, P1NN results were corrupted with neutral probabilities at the random block locations. Table 5.2 shows that the results were not significantly impacted, when the corruption rate of the pixel labeling increases. This supports my claim that the pixel labeling is a practical hint which is robust and is somewhat immune to the noise.

## 5.3 Brodatz Texture Dataset

In the previous section, I showed that classification tasks that take pixel labels as inputs are robust to pixel-labeling errors, and pixel labels provide useful hints to the subsequent tasks. It also suggested that the pixel labeling layer can be generalized to other problems. However, the Pentomino dataset misses an important characteristic that I believe is crucial for classifying histology images. The identification task lacks variation of textures. The Pentomino dataset is monochrome, with no texture information to suggest a Pentomino type. Furthermore, in Pentomino dataset, the aperture window is often only weakly informative of a Pentomino's type. It is less likely that I can find a relationship between the features learned from a window and the identification for a Pentomino type. However, in the histology dataset, I believe that a substantial amount of information is encoded in the context, which can be interpreted as texture in computer vision.

Texture is widely recognized as an the important characteristic used for identifying objects in the images (Haralick et al., 1973). Different techniques have been proposed to solve the grayscale texture problems (Haralick et al., 1973; He and Wang, 1990; Laine and Fan, 1993; Bovik et al., 1990; Huang et al., 2009; Voisin et al., 2013).

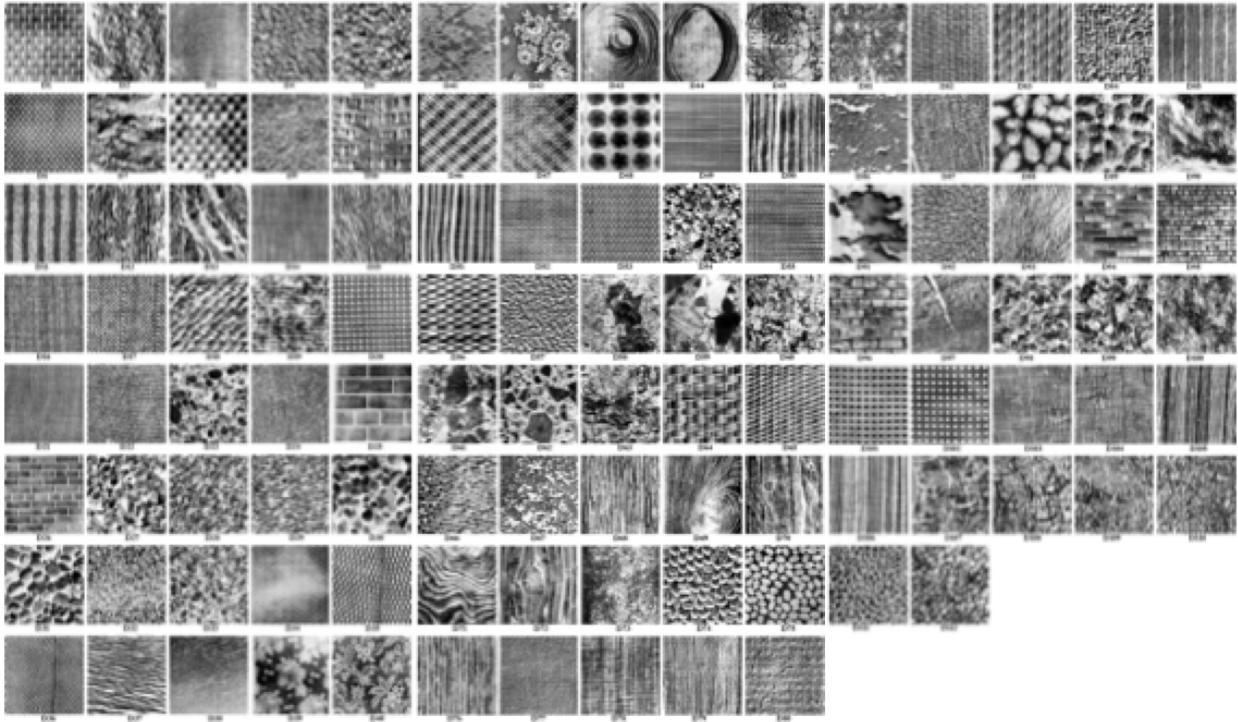
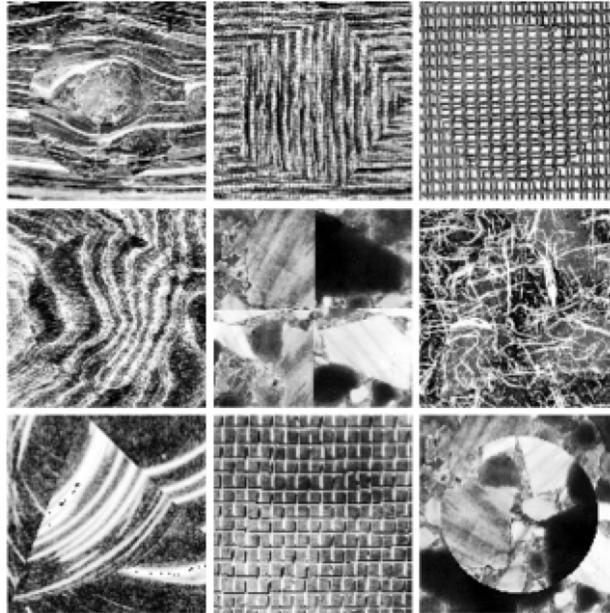


Figure 5.7: Examples of 112 texture images from Brodatz Album.

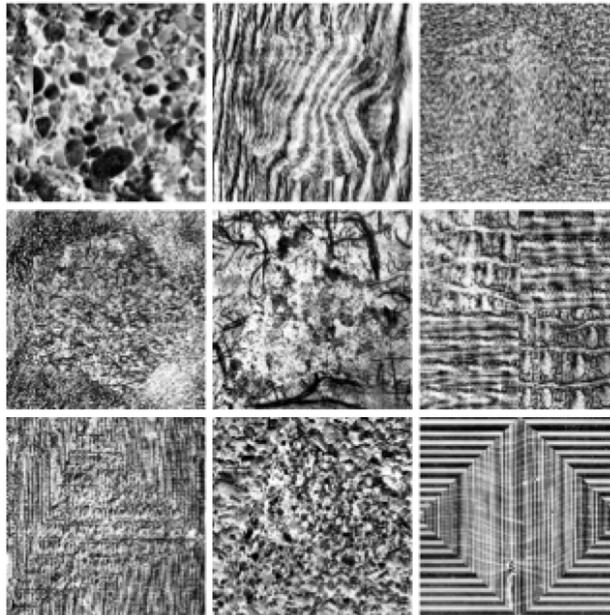
The Brodatz Album (Brodatz, 1966) has become the standard for evaluating the texture algorithms and been widely used as a validation dataset (Liu and Fieguth, 2012; Yang et al., 2012; Díaz-Pernas et al., 2009). It is composed of 112 images representing a variety of natural textures with different background intensities as shown in Figure 5.7. However, some of the texture images could be discriminated using only the mean intensity and variations without texture information. Thus, I chose the Normalized Brodatz Texture (NBT) database, which eliminates the grayscale background effect. I manipulated this dataset to create a toy problem, which is equally hard to learn and can be benefit from pixel labeling.

### 5.3.1 Problem Definition

Figure 5.7 illustrates the 112 texture images in the Brodatz Album. Based on these 112 texture images, I created another toy dataset, where two textures are selected with replacement and assigned as either the foreground or background texture in an image. The ultimate goal of this task was to determine whether the foreground texture is different from the background texture. To make



(a)



(b)

Figure 5.8: (a) Examples from the generated dataset, where their foreground and background textures are from the same texture images. (b) Examples with different foreground and background textures that are hard to distinguish. Foreground and background textures are randomly selected with replacement and the shape within the image is also randomly chosen from circle, pentagon, square, triangle and quarter. Two textures are mixed with roughly the same area for each one.

sure the dataset is not biased, the numbers of examples with the same and different foreground and background textures were equal. Figure 5.8(a) shows the examples where the foreground and background are from the same texture, and Figure 5.8(b) shows the different.

### **5.3.2 Experiment Setup**

To make the toy problem more similar to histology image dataset, I introduced several variable factors, including rotations, mask shapes as well as different locations, to the dataset. This section presents the process of how I generated new texture images based on Brodatz Album, as well as the size and parameter settings included in the process.

The model architecture used in the experiments is also presented in the section. I provided the hints for a region, instead of serving the labeling hints at the pixel level, since we know that generally the labels are coherent within a region. It would not be necessary to keep all pixel labels if most of them are the same. Finally, the labeling hints are provided to the subsequent task to determine whether the background and foreground textures in the image are the same or not.

#### **5.3.2.1 New Texture Image Generation**

This process aims to create a texture dataset which is equally hard to learn as the original histology problem and depends on the detailed information, such as identifying a texture, for an accurate classification. To increase overall complexity and difficulty for the classification, I added three variations to the image generation process, including selecting a random patch from each texture, rotation by one of four different angles, as well as five different foreground/background shapes. Moreover, to make it harder, the textures with similar background effects are grouped together as a set prior to the generating process, and the chances for the textures with similar background intensity are more likely to be paired as the foreground and background textures as shown in Figure 5.8(b).

First, a texture is randomly picked with replacement from Brodatz Album, and the other texture is randomly drawn from a corresponding texture set, which has been grouped based on the

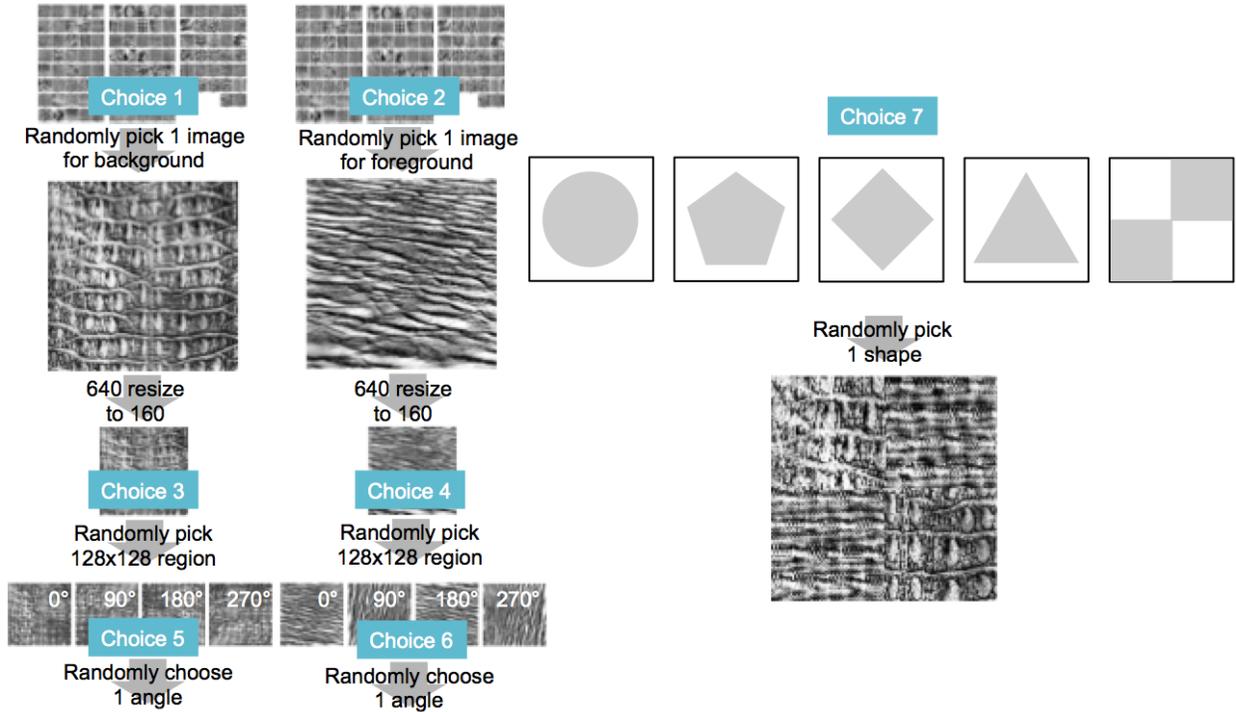


Figure 5.9: The generation process of the new texture dataset. First, two textures are randomly picked from Brodatz Album and assigned as the foreground or background texture. The 128 square regions are then cropped from the resized 160 square area at random locations. The two texture images are rotated by four choices of angles. Finally, the two textures are mixed with roughly the same pixels in a random shape.

background effect in advance. Each of them is assigned as the foreground or background texture for the new image. Then,  $128 \times 128$  regions are cropped from the resized  $160 \times 160$  square area at random locations. After that, the two texture images are rotated by four choices of angles. Then, one shape is designated from five different choices, including circle, pentagon, square, triangle, and quarter. Finally, the two textures are mixed with roughly the same pixels in each one. The numbers of samples of the same and different foreground and background textures were set equal to ensure the dataset is not biased. Figure 5.9 illustrates the generating process of the new texture dataset.

Variable factors, such as different locations, rotations, and mask shapes, introduced to the dataset are designed on purpose. I intend to confuse the model with these variations and make it clueless of what representations are important and what to predict during the learning process. Unless some specific guidance is provided to tell the model what should learn the features of textures

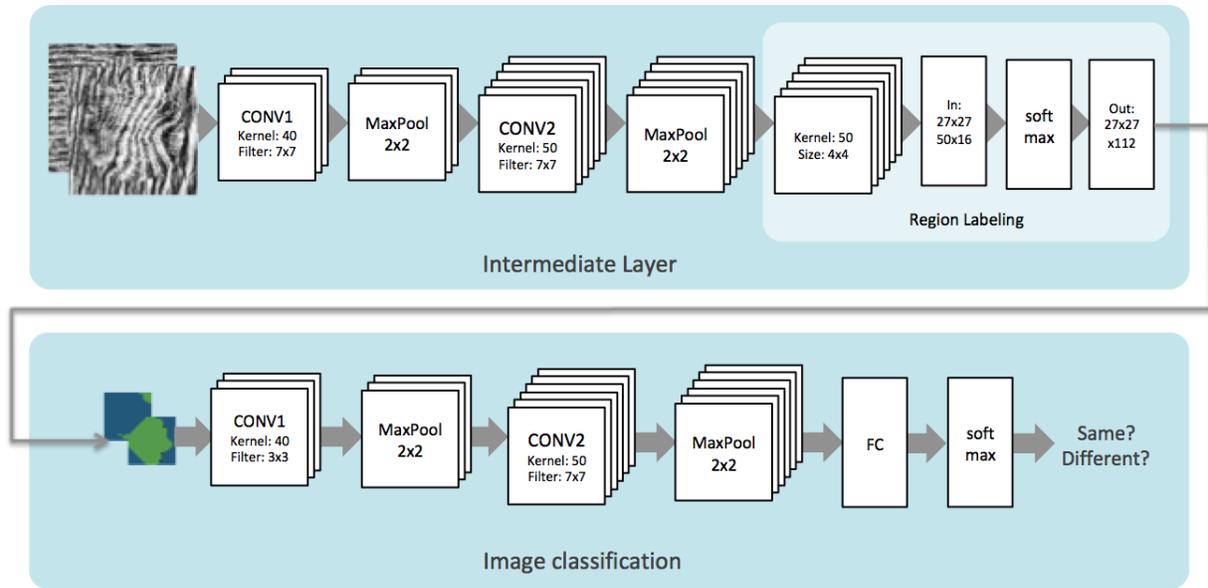


Figure 5.10: Model architecture for region labeling.

first for example, the model might spend time learning the features of irrelevant details embedded in the image.

To evaluate the model, 20,000 examples are generated, 13,280 examples for training and 3,320 examples for testing and validation. Four different rotations and five distinct shapes are equally represented in the dataset.

### 5.3.2.2 Region Labeling Hints

The overall architecture for the region labeling is illustrated in Figure 5.10. I used a standard CNN with four convolutional layers and one fully-connected layer to solve the texture problem. The input of the architecture is the  $128 \times 128$  images I generated, and the output of the model is a binary answer of yes or no. One layer in the architecture is picked and served as the intermediate layer for the region labeling. In this toy problem, I decided to label regions at a lower resolution in the labeling layer instead of every pixel of an original image because the labels are usually coherent. It would be not necessary to keep all pixel labels if they are the same. Region labeling could save both time and space computing this layer.

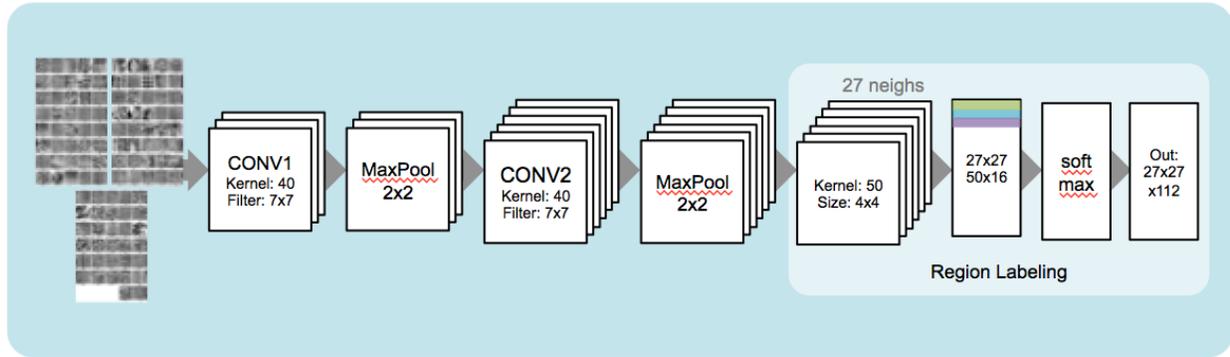


Figure 5.11: Model architecture for supervised region labeling model.

Therefore, I took a window size of  $4 \times 4$  block across kernels and flatten them to be one vector of the softmax layer's input. The process collects  $27 \times 27$  blocks with the window size of  $4 \times 4$  pixels across 50 kernels, and all these  $27 \times 27$  vectors finally are consolidated to be the input of the softmax layer. The output of the softmax layer, which is then fed to the classification model, is the probabilities of labels for the corresponding block. The classification model uses those hints learned from the labeling model and determines whether the image has same or different textures.

The architecture exploits a hint about the texture class, and provides semantic information for the region labeling layer. Region labeling is trained with the intermediate target, specifying the texture class for each region. A hyperbolic tangent nonlinearity was used for the activations in the model, since using  $\tanh(\cdot)$  nonlinearity function makes the optimization more tractable compared to others (Heaton, 2017). The learning rate for the region labeling model was 0.1 with 200 epochs training. I have experimented several different adaptive learning rate algorithms, and found that using noisy gradient descent (Neelakantan et al., 2015) converges faster to local minimum and helps the optimization problem escape from the saddle points. In addition, an early stopping scheme was used to avoid overfitting while training.

### 5.3.3 Experiment Results

To learn the intermediate layer hint in this toy dataset, I first independently trained a region labeling model. While the input images are different, the architecture of the region labeling is

Table 5.3: Classification results on generated texture dataset with different levels hints provision.

Model	Accuracy
(a) CNN without hints / initialized weights	51.386%
(b) CNN without hints + fixed initialized weights	54.096%
(c) CNN without hints + fine-tuned initialized weights	79.699%
(d) CNN with <i>Pixel labeling</i> hints (learned with two different targets)	85.030%

identical as shown in Figure 5.11. The architecture for this experiment uses the images from Brodatz Album, applying rotations and translations but with no shape masks to ensure similar complexity to the generated texture dataset. Using the Brodatz dataset, I know the answer of every pixel of the input image, and the target value for every region in an image would have the same texture label. The model is expected to learn the features for the corresponding textures, and every region in an image should be labeled the same. This is an easy to learn problem that the model can easily identify the textures with around 97% accuracy.

To learn the binary image classification model, whether two features are different or the same, the input of the model is the generated two-texture dataset. When no intermediate hints were applied to the model, I got an accuracy no better than a coin-flip as shown in Table 5.3(a).

Glorot and Bengio (2010) shows that initializing parameters in a deep neural network can have a big impact on the learning and generalization. Erhan et al. (2010) also points out that initialization of parameters worked with pretraining process can guide the learning towards effective local minima with better generalization. Accordingly, I utilized the hints learned from the pretrained supervised region labeling introduced previously by initializing the weights of the region labeling model with the weights learned from Brodatz input *without fine-tuning*, the model have around 55% accuracy as shown in Table 5.3(b).

However, deep neural networks like CNNs involve a huge number of parameters. In practice, it is common to fine-tune the weights of the pretrained CNNs by continuing the backpropagation. Provided that the dataset contains the same context to the original one, the pretrained model already has learned the features that are relevant to the classification problem. Therefore, I initialized

the weights as the first stage and let the model learn from there, the model can have around 80% accuracy as shown in Table 5.3(c) on determining the same or different textures in an image.

### 5.3.4 Optimize Two Problems with Joint Objective Functions

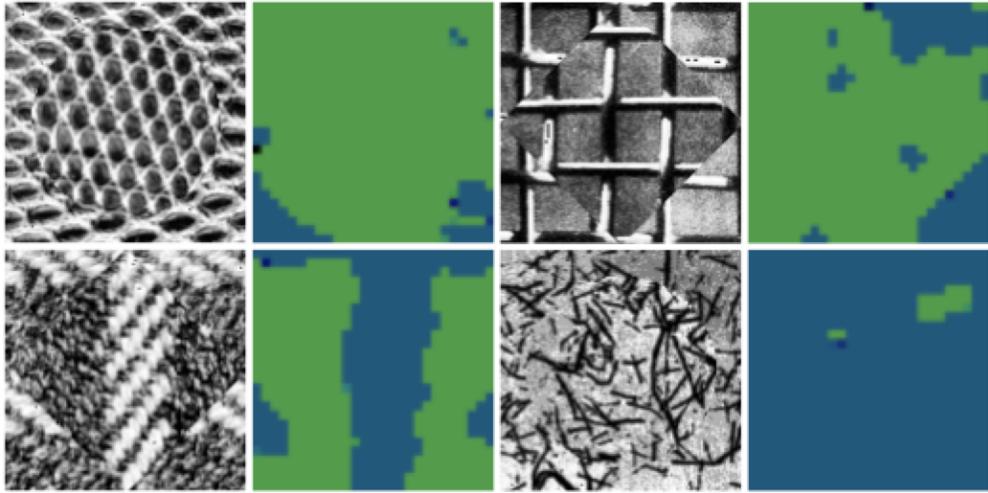
I further trained the image classification with the region labeling hints by jointly optimizing the loss for the labeling hints  $C_{\text{labeling}}$  and for the meta-task  $C_{\text{classification}}$  together as a joint loss:

$$\underset{\theta}{\text{minimize}} \quad w_h C_{\text{labeling}} + (1 - w_h) C_{\text{classification}} \quad (5.1)$$

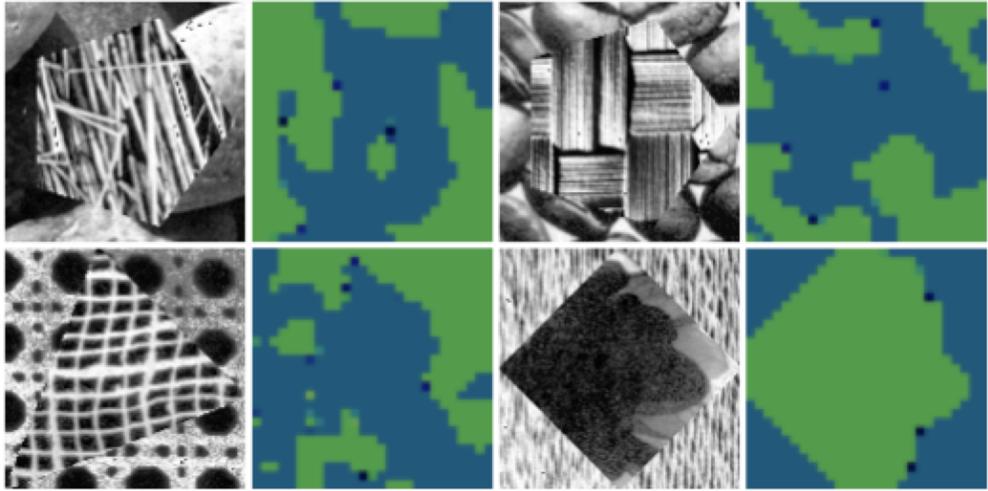
where  $\theta$  represents the set of all parameters and  $w_h \in [0, 1]$  is the guided ratio, which is set to 1 at the beginning of the training and linearly decreased to 0.1 in 10,000 updates. If  $w_h = 0$ , the objective is equivalent to  $C_{\text{classification}}$ , and equals to  $C_{\text{labeling}}$  as  $w_h = 1$ . I found that jointly trained on the hint model and the classification model can improve the ultimate classification results significantly.

The optimization process concentrates on different objectives according to the weight  $w_h$ . Early in the process it focuses on getting the labeling model correct, and as the process goes on, it emphasizes the accuracy of the classification model. I gathered two datasets for both models and let two models learn jointly, and the results can attain higher than 85% accuracy as shown in Table 5.3(d).

Figure 5.12 presents some samples from the generated Brodatz dataset and their corresponding false-color results indicating the label with the highest probability at each pixel. Figure 5.12(a) are the examples with the same foreground and background textures. From the observation of the false-color images, although the model has been misled by some distinct features represented in the same image which results in identifying two different labels, the model is able to recognize different textures for most samples. On the other hand, Figure 5.12(b) shows the examples with different textures for the foreground and background. The labeling model generally can identify different textures and the borders between the textures in some examples are clearly discovered.



(a)



(b)

Figure 5.12: (a) Samples from generated dataset with the same foreground and background textures and their false-color images indicating the label with the highest probability at each pixel. (b) Samples and their false-color images with different foreground and background textures.

As a result, the experiments presented in this section suggest that a hard to learn problem can be made simple if we can guide the model to learn in the right direction through curriculum learning. It also shows that a labeling model can be generalized to different dataset.

## 5.4 Conclusion

In this chapter, I applied my proposed labeling model to two different datasets, a classic Pentomino dataset and a new Brodatz texture dataset, to demonstrate the generalization of the labeling model I have developed. In the Pentomino toy problem, the experiments show that my labeling model can provide meaningful hints to different subsequent tasks, P1NN and P2NN, and improve the performance significantly. Although the pixel labeling hints are not as good as SMLP-Hints in 40k dataset, they still proved to be useful and a more practical classifier in the real world. Also, the sensitivity experiments prove that the pixel labeling classifier is robust and not susceptible to the noise.

In addition, I presented a new toy problem based on Brodatz texture Album, which, I believe, is more similar to the histology image scenario. The experiments imply that the labeling model can provide valuable hints for the ultimate classification goal. Even if the model is not directly learned with hints but initialized with the weights learned from the supervised labeling model, the accuracy can still be increased by 30%. Furthermore, optimizing the model by jointly learning the labeling hints and the meta-task improves the performance even more significantly.

In conclusion, I evaluated the labeling approach on two toy problems, the experiments suggest that the advantages of a pixel labeling classifier are threefold. First, it generalizes to other related vision tasks. The experiments in two toy problems show that the classification performance is improved with the labeling hints. Second, it is robust and not susceptible to pixel label noise. The labeling approach serves as an intermediate hint for the meta task. Image classification does not require extremely accurate pixel labeling. Third, it is beneficial for classifying a texture-like dataset. The labeling model uses a small aperture window to analyze the elements composed in an object.

The low-level texture information might be lost while training goes deeper in the hierarchy, but the labeling model captures them and converts them to valuable hints.

## CHAPTER 6: UNSUPERVISED PIXEL LABELING LAYER

### 6.1 Introduction

In Chapter 4, pixel/region labeling proved to be useful and practical as an intermediate hint, and I also showed that it generalizes to different problems in Chapter 5. Still, one might argue that the need for an extra training dataset for the pixel labeling model makes the hint augmented training less appealing. This observation motivates the question of whether is there a way to construct a labeling model with the benefits of the hint layer, but without explicitly providing hints.

An unsupervised labeling model might make this additional training process unnecessary. First, because no ground-truth labels are required during training process, no extensive prior knowledge is required. Second, the hint labels are generated purely based on the data, and therefore they are not subjective to visual interpretation (Bengio, 2012; Berg-Kirkpatrick et al., 2010; Xie et al., 2016; Dizaji et al., 2017).

However, optimizing an unsupervised deep learning model is challenging. The model has to learn underlying feature representations and assign clusters simultaneously. Without additional labeled training data, the model has to iteratively refine the label assignments with an auxiliary learning objectives for shaping the output distribution to gradually improve the labeling results as well as refine the feature representations (Bengio, 2012; Xie et al., 2016).

In this chapter, I propose an approach which extends the architecture introduced in Chapter 4. The classification task is divided into two subtasks, a low-level labeling and an image-level classification. The labeling model is trained by a CNN, whose output is a probability of labels. The output of the labeling model serves as an intermediate hint and is used as the input of image classifier for further training.

I aim to investigate the potential for an unsupervised learning objective which includes three regularization terms. The terms are inspired by the observations of the supervised labeling results, and are designed to constrain the shape of the output distribution. They implicitly aid the unsupervised model in learning feature representations and ultimately output cluster-like labels for the downstream learning task.

## 6.2 Approach

In this section, I introduce three penalization terms to the output of the labeling model based on the desired property of its probability distribution. The approach is studied and evaluated on Brodatz texture dataset introduced in Section 5.3. Section 6.2.1 examines the importance of the probability distribution of the pixel labeling output to the unsupervised loss function. In addition, inspired by an approach proposed by Kilinc and Uysal (2017), I present three regularization terms, uncertainty, sparsity, and distinction, to form the learning objective for the unsupervised labeling model. These terms enforce or penalize *correlation* relationships between label distributions throughout the unsupervised learning, which is described in Section 6.2.2. Finally, the details of three regularization terms along with the relationships to other measures are detailed in Section 6.2.3.

### 6.2.1 Probability Distribution

The improvements on the Brodatz toy problem with supervised intermediate hints, which was introduced in Section 5.3.3, motivates the question of what characteristics does the supervised labeling model have that leads to this progress, and what information does it indicate.

I examined the probability distribution of softmax output in the labeling model over all pixels in a region as illustrated in Figure 6.1. The false-color image in Figure 6.1 is a softmax output in the labeling model with same foreground and background textures indicating the label with the highest probability at each pixel. Each pixel outputs a corresponding probability vector over all texture labels. A cumulated probability distribution illustrated in Figure 6.1 is a distribution consolidating

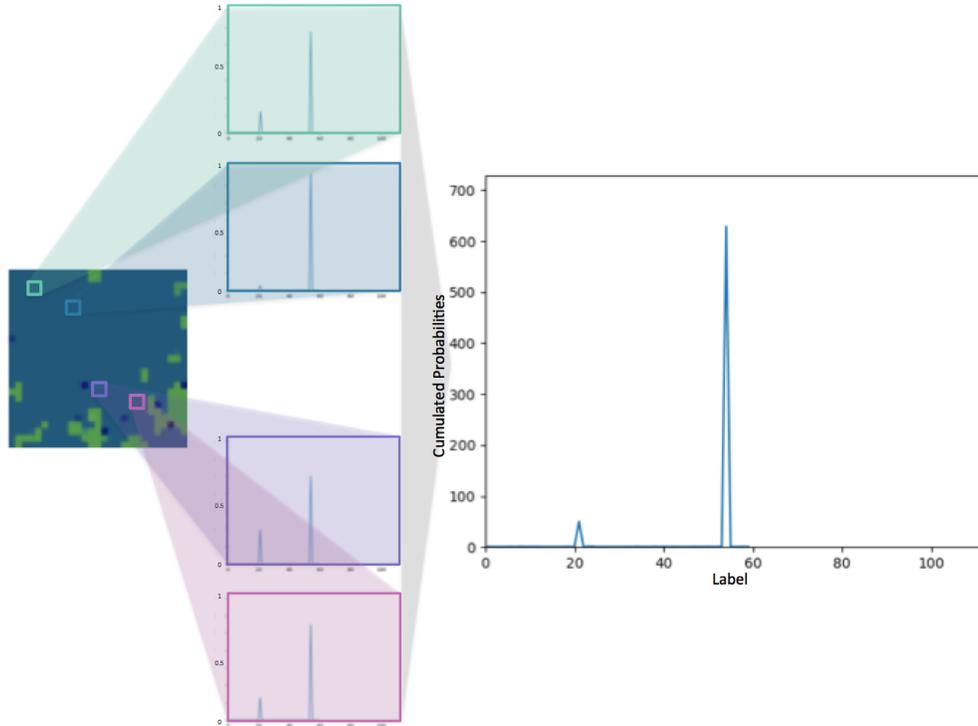


Figure 6.1: An example of cumulated probability distribution over 729 ( $27 \times 27$ ) pixels in the region. The x-axis represents 112 labels. The y-axis is the probability sum over all pixels in the region, where 729 is the maximum value when the probabilities of all pixels are 1.

the probability of each texture label over all pixels. The x-axis represents 112 texture labels, and the y-axis is the probability sum over all pixels in the region, and it reaches 729 when the probabilities of all pixels are 1.

Three important attributes arise from this examination of models learned via supervised-learning. First, identifying how many significant textures are found in each region? Second, how certain is the model's prediction of the pixel labels? Third, does the model accurately identify the right textures? I find that once a model is learned, the two measures tend to exhibit profiles that are highly predictive of whether the foreground and background textures are the same. Figure 6.2(e) and (f) show cumulative probability distributions of the examples with same and different foreground and background textures, respectively. Figure 6.2(e) shows the maximum value on label 22, indicating that the model perfectly recognizes all pixels the same texture and is very confident about the predictions for all pixels. On the other hand, Figure 6.2(f) has two peaks on label 21 and 53,

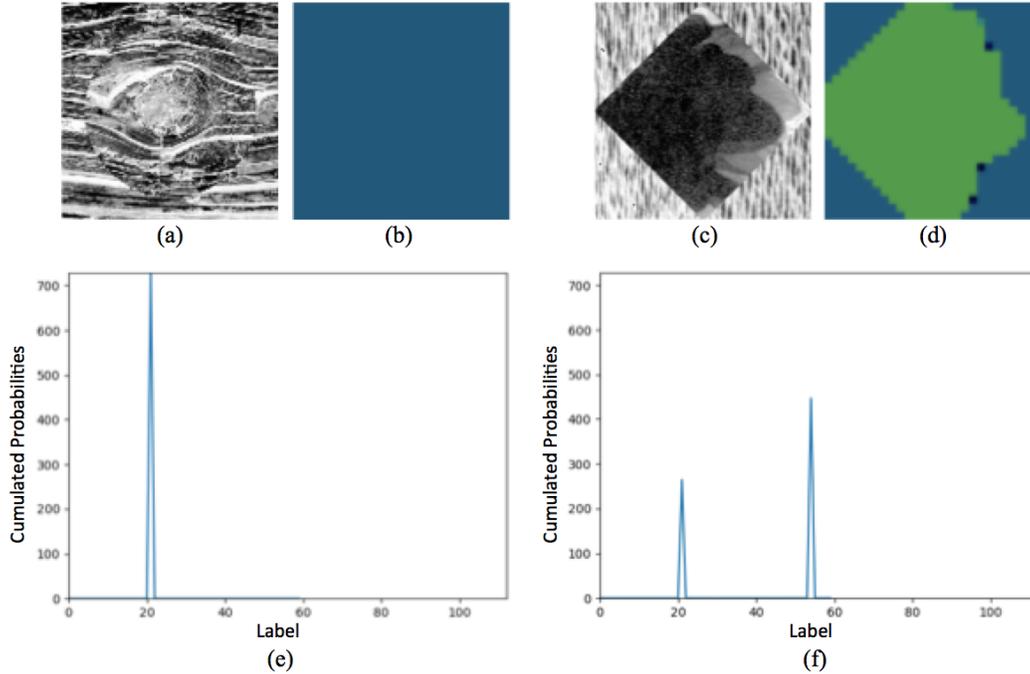


Figure 6.2: (a)(c) are the examples of same and different foreground and background textures, respectively, in the generated texture dataset. (b)(d) are the false-color images indicating the label with the highest probability at each pixel. (e)(f) are the cumulated probability distribution over all pixels in the region.

respectively, showing that the model is certain that there are two different textures appearing in the image.

From this observation, it is possible to learn if the sample has the same foreground and background texture, the corresponding cumulative probability distribution will have one peak, and two peaks otherwise. Accordingly, to construct the unsupervised loss function for the labeling model, I assume that a good labeling output should have following constraints.

1. The label at each pixel should be as unambiguous as possible.
2. There are relatively few different textures within a window.
3. All textures are represented uniformly within a randomized mini-batch.

First, the model ideally predicts only one label for every pixel; that is, for every pixel in one sample, it should be assigned to one texture with high probability, while remaining textures have

very low probabilities. Second, there should be relatively few labels appeared within a window. For example, within a given  $5 \times 5$  window, it is unlikely that 25 different textures appear in it; instead, the number of the labels should relatively small because the labels in the nearby pixels are generally coherent. Lastly, these two properties alone are not sufficient to discover distinct texture labels. The first two properties can fulfill the requirements of one peak for same textures and two peaks otherwise, but they can not guarantee that peaks map to different labels.

Here, I use a common training mechanism for learning deep models, mini-batch gradient descent (Li et al., 2014), to learn a solution. Mini-batch gradient descent is a variation of the gradient descent algorithm that is widely used in deep learning model. It splits the training dataset into small mini-batches which are then used to calculate the model error and update model coefficients. I assume that within a mini-batch textures are represented uniformly. This constraint helps the model associate different peaks to different textures.

In conclusion, my goal for the unsupervised labeling model is to construct a cumulative probability distribution which can satisfy all three of these constraints. If they are all satisfied, the labeling model at the pixel level can be sure in one texture, predict as few textures as possible locally within a window, and distinguish different textures within a mini-batch.

## **6.2.2 Correlation Relationships**

A correlation is an analysis that measures the strength of relationship between two variables and the direction of the relationships. The value of correlation coefficient is between -1 and +1 in terms of the degree of the relationship. When the value of correlation lies around +1, it is said that two variables are positively correlated, meaning two variables increase or decrease together. On the other hand, when the correlation coefficient is negative, it indicates the extent to which one variable increases as the other decreases. As the correlation coefficient value goes towards 0, the relationship between the two variables becomes weaker.

Geometrically, correlation is related to the cosine of the angle between two high-dimensional feature vectors. A dot product, which is a common component of correlation, calculates the cosine

of the angle between two vectors multiplied by their magnitudes:

$$\langle x \cdot y \rangle = \cos\theta \|x\| \|y\| \quad (6.1)$$

To find the relationship between correlation and cosine, I first recall that the covariance is calculated as follows:

$$\text{Cov}[X, Y] = E[XY] - E[X]E[Y], \quad (6.2)$$

and correlation is defined as

$$\text{Corr}[X, Y] = \frac{\text{Cov}[X, Y]}{\sigma[X]\sigma[Y]} \quad (6.3)$$

If one of the random variable has expected value of 0,  $\text{Cov}[X, Y] = E[XY]$  is the dot product of  $X$  and  $Y$ . The standard deviation of the variable is the length of itself. The correlation is then the cosine of the angle between two vectors. Accordingly, a positive correlation shows an acute angle. The negative correlation means an obtuse angle, and uncorrelated is orthogonal, which correlation coefficient is 0.

Based on this, if there are two variables uncorrelated, namely correlation coefficient is 0, the dot product of two vectors should be 0 correspondingly. Therefore, I propose to apply regularizations during the unsupervised task in order to make sure that the dot product of two vectors is minimized if two vectors are not correlated.

### 6.2.3 Loss Function Definition

In this section, I introduce three regularization terms, uncertainty, sparsity, and distinction, to form the loss function for the unsupervised labeling model. Each of them is designed to constrain the predicted output to preserve three desired properties listed in Section 6.2.1. The details of three regularization terms along with the relationships to other measures are detailed in the following sections. Lastly, a scheme of scheduling weights is exploited. It controls the importance between the loss functions of the labeling model and the meta classification model during the training time.

More details on how this approach engages in the joint learning process of two models is described in the last part of this section.

### 6.2.3.1 Uncertainty

**Constraint 1:** *Assuring that the label at each pixel is unambiguous. Ideally, every pixel in a sample would be assigned to one texture with the probability of 1, while the remaining textures would have 0 probability.*

To ensure that the output of the trained labeling model satisfies the constraint 1, I designed a penalty which can be used to regularize the model's training objective. Suppose that after training, the predicted matrix of labeling model  $P$  is obtained, where  $P$  is a  $m \times n$  matrix with  $m$  pixels within a window and  $n$  texture labels.  $P_{ij}$  is the probability of the  $i$  sample belonging to  $j$  label. I assume that if the label at one pixel is not ambiguous, the probability vector for one texture label should be as uncorrelated to the other as possible. Based on the correlation relationships introduced in Section 6.2.2, the dot product of two probabilities vectors should go 0 if they are uncorrelated. Therefore, I derived the matrix  $C$ , a surrogate of true correlation matrix, by taking the predicted output matrix  $P$  dot product by itself:

$$C = P^T P = \begin{bmatrix} \sum_m P_{i1}P_{i1} & \sum_m P_{i1}P_{i2} & \dots & \sum_m P_{i1}P_{in} \\ \sum_m P_{i2}P_{i1} & \sum_m P_{i2}P_{i2} & \dots & \sum_m P_{i2}P_{in} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_m P_{in}P_{i1} & \sum_m P_{in}P_{i2} & \dots & \sum_m P_{in}P_{in} \end{bmatrix}, \quad (6.4)$$

and  $C$  is a  $n \times n$  symmetric matrix, where  $C_{ij}$  denotes the correlation between two labels. For this regularization term, I force the matrix  $C$  to become a diagonal matrix, where 0 for the correlation coefficient of two different labels and positive values on the diagonal. It is noteworthy that  $P_i$  is

the probability vector of label  $i$  over all pixels, and thus  $C_{ii}$  represents the sum of the square of probabilities for label  $i$  over all pixels.

Inspired by the work presented in Kilinc and Uysal (2017), I exploited a regularization term, *uncertainty*, to penalize the non-zero off-diagonal entries of  $C$  to constrain  $C$  to be a diagonal matrix. The term is defined as

$$\text{uncertainty}(P) = \frac{\sum_{i \neq j}^n C_{ij}}{(n-1) \sum_{i=1}^n C_{ii}} \quad (6.5)$$

The uncertainty score approaches 0 in an ideal classified block. Normalization in uncertainty score is done to bring different regularization terms within the same range for optimization, and ultimately simplify hyperparameter adjustment. During the optimization, uncertainty score is expected to be minimized to ensure that the constraint holds true.

In the following I give three examples to make the idea more concrete. Let  $P$  be the predicted softmax output matrix of labeling model, with five pixels and three texture labels. In the examples shown below, row vectors are pixel samples, and columns are the texture labels. The values in each row sum to 1. If the probabilities are uniformly distributed, indicating that the three textures are equally likely, then the uncertainty score is high.

$$\text{uncertainty}\left(\begin{bmatrix} 0.34 & 0.33 & 0.33 \\ 0.33 & 0.34 & 0.33 \\ 0.33 & 0.33 & 0.34 \\ 0.33 & 0.33 & 0.34 \\ 0.34 & 0.33 & 0.33 \end{bmatrix}\right) = 0.9997, \quad C = \begin{bmatrix} 0.56 & 0.55 & 0.55 \\ 0.55 & 0.55 & 0.55 \\ 0.56 & 0.55 & 0.56 \end{bmatrix}$$

However, if the predict model tends to show a preference for a specific texture, as indicated by a higher probability on those textures, the uncertainty score starts dropping.

$$\text{uncertainty}\left(\begin{bmatrix} 0 & 0.1 & 0.9 \\ 1 & 0 & 0 \\ 0.2 & 0.8 & 0 \\ 0.3 & 0 & 0.7 \\ 0 & 0.7 & 0.3 \end{bmatrix}\right) = 0.1831, \quad C = \begin{bmatrix} 1.13 & 0.16 & 0.21 \\ 0.16 & 1.14 & 0.30 \\ 0.21 & 0.30 & 1.39 \end{bmatrix}$$

And lastly, if all pixels have the probability 1 is for the certain textures and 0 for the rest, the uncertainty score would be 0. Note that if we reach this extreme case, the diagonal of  $N$  is the number of pixels been predicted as the corresponding texture.

$$\text{uncertainty}\left(\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) = 0, \quad C = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

From the observation, the lower uncertainty score it gets, the higher probabilities the prediction model gives to the certain labels. As a result, if I minimize the uncertainty score during the learning process, the model can ultimately fulfill the constraint 1. Uncertainty is minimized when every pixel is certain of its label.

### 6.2.3.2 Sparsity

**Constraint 2:** *There are relatively few different textures within a window.*

Recall that  $C$  is a  $n \times n$  symmetric matrix, such that  $C = P^T P$ , which represents the correlation relationships among labels from the labeling predicted output  $P$ . The diagonal entry  $C_{ii}$  is the sum square of probabilities for label  $i$  over all pixels. Because of that, I can tell how many labels have significantly high probabilities from interpreting the diagonal entries of  $C$ , which implies the number of distinct labels been identified within a window. For constraint 2, I expect the labeling model to identify fewer distinct textures within a window, because generally the neighbors have

the same texture within a region. Thus, the correlation matrix  $C$  should not only be forced to be a diagonal matrix, but also have as few distinct high values on the diagonal as possible.

Here, I apply the same trick used for regularizing constraint 1 to correlation matrix  $C$ . Let  $v$  be a  $1 \times n$  vector representing the diagonal entries of  $C$  such that  $v = [C_{11} \ C_{22} \ \dots \ C_{nn}]$ , and let  $V$  be a  $n \times n$  symmetric matrix such that

$$\begin{aligned}
 V &= v^T v \\
 &= \begin{bmatrix} C_{11}C_{11} & C_{11}C_{22} & \dots & C_{11}C_{nn} \\ C_{22}C_{11} & C_{22}C_{22} & \dots & C_{22}C_{nn} \\ \vdots & \vdots & \ddots & \vdots \\ C_{nn}C_{11} & C_{nn}C_{22} & \dots & C_{nn}C_{nn} \end{bmatrix} \\
 &= \begin{bmatrix} \sum_{i=1}^m P_{i1}^2 \sum_{i=1}^m P_{i1}^2 & \sum_{i=1}^m P_{i1}^2 \sum_{i=2}^m P_{i2}^2 & \dots & \sum_{i=1}^m P_{i1}^2 \sum_{i=n}^m P_{in}^2 \\ \sum_{i=2}^m P_{i2}^2 \sum_{i=1}^m P_{i1}^2 & \sum_{i=2}^m P_{i2}^2 \sum_{i=2}^m P_{i2}^2 & \dots & \sum_{i=2}^m P_{i2}^2 \sum_{i=n}^m P_{in}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=n}^m P_{in}^2 \sum_{i=1}^m P_{i1}^2 & \sum_{i=n}^m P_{in}^2 \sum_{i=2}^m P_{i2}^2 & \dots & \sum_{i=n}^m P_{in}^2 \sum_{i=n}^m P_{in}^2 \end{bmatrix}
 \end{aligned} \tag{6.6}$$

It is obvious that if the value of  $C_{ii}$  is small, all of the off-diagonal entries in  $V$  which are multiplied by  $C_{ii}$  will conversely, if I compel the off-diagonal entries in  $V$  which are multiplied by  $C_{ii}$  to be small, the value of  $C_{ii}$  or  $C_{jj}$  would correspondingly get smaller. When more diagonal entries in  $C$  get smaller, it would be highly likely that the number of labels with significant high probabilities becomes fewer. Therefore, a regularization term, *sparsity*, is then proposed to penalize the non-zero off-diagonal entries of  $V$ :

$$\text{sparsity}(P) = \frac{\sum_{i \neq j}^n V_{ij}}{(n-1) \sum_{i=1}^n V_{ii}} \tag{6.7}$$

Sparsity score falls between 0 and 1, where 0 represents the case that off-diagonal entries in  $V$  are all 0 and only one value is not 0 on the diagonal entries of  $C$ . Therefore, the unsupervised labeling model should learn to minimize the sparsity score during the optimization.

Here, I present the examples in different cases for better understanding.  $P$  are the probabilities of different labels of the labeling model with ten pixels and three texture labels, and  $C$  is the correlation matrix for the probabilities of labels in  $P$ .  $V$  is the new defined matrix, which is the dot product results of diagonal entries in  $C$ .

The first example shows an predicted output with eight different textures almost uniformly distributed, meaning the pixels within this window are almost equally among these eight textures. In this case, the sum of off-diagonal entries in  $V$  would be large, and hence its sparsity score would be very high.

$$\text{sparsity} \left( \begin{bmatrix} 0.832 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 \\ 0.012 & 0.916 & 0.012 & 0.012 & 0.012 & 0.012 & 0.012 & 0.012 \\ 0.9 & 0.9 & 0.937 & 0.9 & 0.9 & 0.9 & 0.9 & 0.9 \\ 0.6 & 0.6 & 0.6 & 0.958 & 0.6 & 0.6 & 0.6 & 0.6 \\ 0.024 & 0.024 & 0.024 & 0.024 & 0.832 & 0.024 & 0.024 & 0.024 \\ 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \\ 0.024 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 & 0.832 & 0.024 \\ 0.030 & 0.030 & 0.030 & 0.030 & 0.030 & 0.030 & 0.030 & 0.790 \\ 0.811 & 0.027 & 0.027 & 0.027 & 0.027 & 0.027 & 0.027 & 0.027 \\ 0.024 & 0.832 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 \end{bmatrix} \right) = 0.90$$

If the predict model recognizes fewer distinct textures, some 0s will start showing up on the diagonal of the matrix  $C$ . Accordingly, there will be some 0s showing in the off-diagonal of  $V$ , and hence the sparsity score becomes lower.

$$\text{sparsity} \left( \begin{bmatrix} 0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 \\ 0 & 0.8 & 0.2 & 0 & 0 & 0 & 0 & 0 \\ 0.1 & 0 & 0 & 0 & 0.9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.2 & 0 & 0 & 0.8 & 0 \\ 0 & 0 & 0.9 & 0 & 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.6 & 0 & 0 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.75 & 0 & 0 & 0.25 \\ 0 & 0.8 & 0 & 0.2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.9 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0.25 & 0 & 0 & 0 & 0.75 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0.8 & 0 & 0 \end{bmatrix} \right) = 0.65$$

$$\text{sparsity} \left( \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \right) = 0.5$$

$$\text{sparsity} \left( \begin{bmatrix} 0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 \\ 0 & 0.8 & 0.2 & 0 & 0 & 0 & 0 & 0.0 \\ 0.1 & 0.9 & 0 & 0 & 0 & 0 & 0 & 0.0 \\ 0 & 0.8 & 0 & 0.2 & 0 & 0 & 0 & 0.0 \\ 0 & 0.9 & 0 & 0 & 0 & 0.1 & 0 & 0.0 \\ 0 & 0 & 0 & 0 & 0.8 & 0.2 & 0 & 0.0 \\ 0.75 & 0 & 0 & 0 & 0 & 0 & 0 & 0.25 \\ 0.8 & 0 & 0 & 0.2 & 0 & 0 & 0 & 0.0 \\ 0.9 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0.0 \\ 0 & 0.75 & 0 & 0 & 0 & 0.25 & 0 & 0.0 \\ 0.8 & 0 & 0 & 0 & 0.2 & 0 & 0 & 0.0 \end{bmatrix} \right) = 0.22$$

Finally, if all pixels are identified as same texture, there would be only one value on the diagonal of  $V$ , and it would get 0 for the sparsity score.

$$\text{sparsity}\left(\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}\right) = 0$$

For this constraint, the value of probabilities in the output matrix  $P$  is not our focus; instead, the number of distinct textures being identified is the crucial indicator impacting the sparsity score most. The lower sparsity score it gets, the fewer distinct textures it will be represented in the matrix  $C$ . In conclusion, for constraint 2, the unsupervised labeling model should attempt to minimize the sparsity score during the optimization process.

### 6.2.3.3 Relationship Between Uncertainty and Sparsity

As mentioned, the uncertainty score is used to estimate how certain the labeling model predicts the labels for a pixel. The predicted probability distribution of labels plays an important role in the uncertainty score. In contrast, the sparsity score is influenced more by the number of significant textures within a window than the probability distribution of labels.

To see how uncertainty and sparsity are related, I conducted an experiment on 320 samples with eight pixels and eight labels within a window, assigning each label with probability from 0.16 to 1.0, where 0.125 would be exactly uniform distribution. 320 samples are divided to eight groups, where the first group has one significant texture label, the second one has two labels, and so on. Following matrices illustrate the examples of the first three groups.

$$P_1 = \begin{bmatrix} 0.58 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 \\ 0.58 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 \\ 0.58 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 \\ 0.58 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 \\ 0.58 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 \\ 0.58 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 \\ 0.58 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 \\ 0.58 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 & 0.06 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 0.79 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.79 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.79 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.79 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.79 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.79 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.79 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.79 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.79 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \\ 0.03 & 0.79 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 & 0.03 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 0.832 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 \\ 0.024 & 0.832 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 \\ 0.024 & 0.024 & 0.832 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 \\ 0.832 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 \\ 0.024 & 0.832 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 \\ 0.024 & 0.024 & 0.832 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 \\ 0.832 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 \\ 0.024 & 0.832 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 \\ 0.024 & 0.024 & 0.832 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 \\ 0.832 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 & 0.024 \end{bmatrix}$$

The results of the uncertainty and sparsity score for these samples are shown in Figure 6.3. Since the probability of texture starts from 0.16, it is almost identical to a uniform distribution. Its uncertainty score is correspondingly high. The uncertainty score finally decreases to 0 when the probability of the significant texture reaches 1. It is noteworthy that when the probability of significant texture is larger than around 0.4, the uncertainty score decreases below 0.5, which is shown as the red line in Figure 6.3.

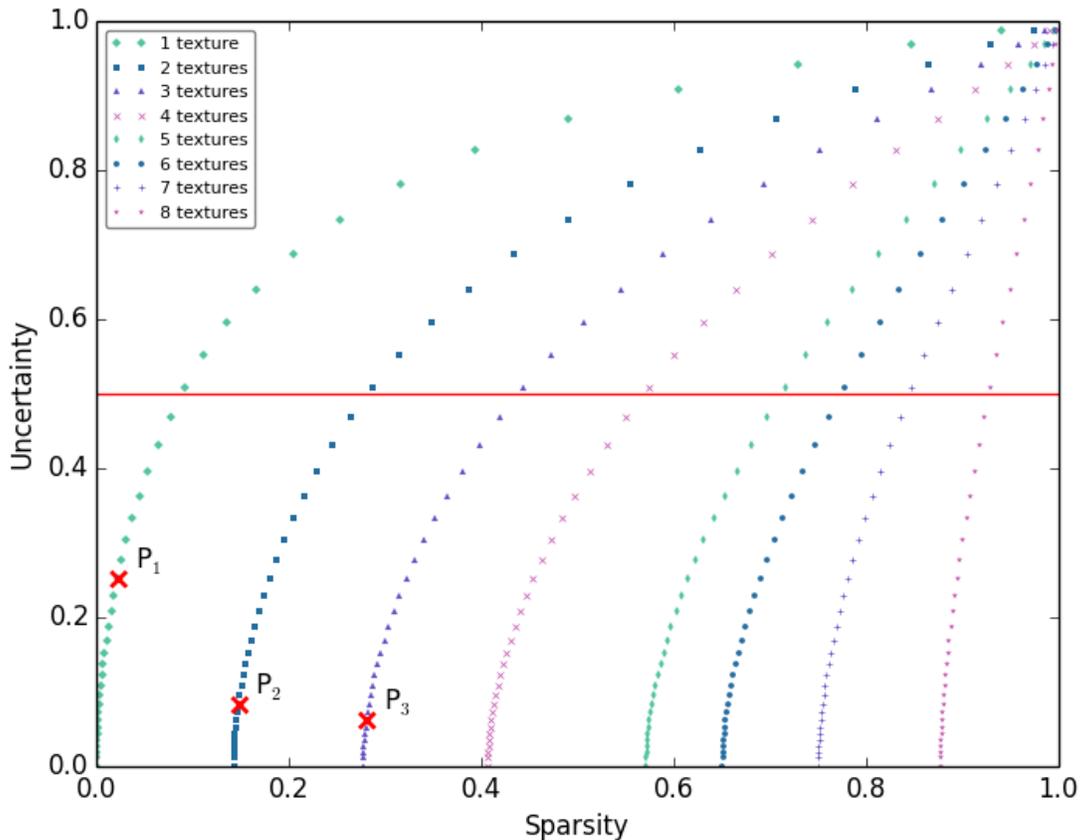


Figure 6.3: Comparison between the uncertainty and sparsity score with 320 examples. Each example contains 10 pixels within a window, showing eight significant textures appearing within a window. The x-axis represents the sparsity score while the y-axis shows the uncertainty score. The red line indicates uncertainty = 0.5. Three examples,  $P_1$ ,  $P_2$ , and  $P_3$  are marked as red crosses in the image. Different colors and glyphs represent various numbers of textures appeared in a window. When the uncertainty scores are less than 0.5, samples with different numbers of significant textures appeared in a window can be clearly identified by their sparsity scores. For example, if the samples have only one significant texture appeared in a window, the sparsity scores fall between 0 and 0.15, samples with two significant textures have the sparsity scores between 0.15 and 0.28, and etc.

On the other hand, Figure 6.3 suggests that the sparsity score can clearly identify the range for different numbers of significant textures appeared in a window, if their uncertainty scores are less than 0.5. For those samples whose sparsity score are overlapped and shown above the red line are the ones with less than 0.4 probability for the significant textures. It is understandable that they are overlapped, because the texture with probability less than 0.4 technically cannot be considered significant. For example, if the probability distribution of first pixel is

like  $[0.16, 0.12, 0.12, 0.12, 0.12, 0.12, 0.12, 0.12]$  and  $[0.12, 0.16, 0.12, 0.12, 0.12, 0.12, 0.12, 0.12]$  for the second pixel, we might regard them as two significant textures, but the fact they are more like a uniform distribution, and should be considered as eight significant textures. In this case, the sparsity score will therefore become higher. Generally, a well learned model with uncertainty score less than 0.5 can effortlessly identify the numbers of significant textures within a window by using a sparsity score.

Regarding to the Brodatz texture dataset, I expect that the model is certain about the predictions, which should have low uncertainty score less than 0.5. In addition, it is reasonable that the model identifies one to three distinct textures within a window, where the sparsity score would fall between 0 and 0.4. Both the distinction and sparsity scores are important to the model training. To keep the scores balance, the coefficients for different scores are introduced to the model, more details discussed in Section 6.2.3.6.

#### **6.2.3.4 Distinction**

**Aim 3:** *All textures are assumed to be represented uniformly within a randomized mini-batch.*

By minimizing the uncertainty and sparsity scores, the unsupervised labeling model is able to constrain the predicted output to have unambiguous labels at one pixel, and few labels within a window. However, only these two constraints are not sufficient to help model distinguish different texture labels. The first two constraints can restrict the shape of the output distribution, but they can not guarantee distinct peaks for different labels. The third assumption is therefore proposed to regularize different output distribution for distinct textures. Although this is the most arguable assumption, it is required to prevent the model from predicting the same output distributions for different textures.

Here, I utilize mini-batch gradient descent, which is a common training mechanism applied widely in deep learning process, to help solve the problem. Mini-batch gradient descent is a variation

of stochastic gradient descent, and it takes more than one training example to make each estimate of the gradient (Li et al., 2014). The training dataset is split into small mini-batches, and each of them is then used to calculate the error and update the model coefficients. I constraint that if  $n$  textures are equally represented in a mini-batch and the model will be forced to output  $n$  distinct labels within a mini-batch, thus allowing it to learn distinct features for  $n$  labels. Even if the batches textures are requested non-uniformly, across multiple mini-batches, using a small step size might achieve the same effect.

Recall that the sparsity score was designed to minimize the number of texture labels showing significant probabilities in the prediction matrix. However, within a randomized batch of samples I expect all textures to be equally likely.

Therefore, I adapted the approach used for sparsity score to a new regularization term, called *distinction*, and try to maximize it within a mini-batch while minimizing sparsity of each example. I do this by summing up the output predicted matrix over all samples in a mini-batch, normalize it, and obtain a new probability matrix  $B$ , where  $B$  is a  $b \times n$  matrix with  $b$  samples within a mini-batch and  $n$  texture labels. I then derive a correlation matrix  $C_b$  by taking the batch matrix  $B$  dot product by itself, and get  $C_b = B^T B$ . Let  $v_b$  be a  $1 \times n$  vector representing the diagonal entries of  $C_b$  such that  $v_b = [C_{b_{11}} \ C_{b_{11}} \ \dots \ C_{b_{nn}}]$ .  $V_b$  is then defined as a  $n \times n$  symmetric matrix, such that

$$\begin{aligned}
V_b &= v_b^T v_b \\
&= \begin{bmatrix} C_{b_{11}} C_{b_{11}} & C_{b_{11}} C_{b_{22}} & \dots & C_{b_{11}} C_{b_{nn}} \\ C_{b_{22}} C_{b_{11}} & C_{b_{22}} C_{b_{22}} & \dots & C_{b_{22}} C_{b_{nn}} \\ \vdots & \vdots & \ddots & \vdots \\ C_{b_{nn}} C_{b_{11}} & C_{b_{nn}} C_{b_{22}} & \dots & C_{b_{nn}} C_{b_{nn}} \end{bmatrix} \\
&= \begin{bmatrix} \sum_b B_{i1}^2 \sum_b B_{i1}^2 & \sum_b B_{i1}^2 \sum_b B_{i2}^2 & \dots & \sum_b B_{i1}^2 \sum_b B_{in}^2 \\ \sum_b B_{i2}^2 \sum_b B_{i1}^2 & \sum_b B_{i2}^2 \sum_b B_{i2}^2 & \dots & \sum_b B_{i2}^2 \sum_b B_{in}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \sum_b B_{in}^2 \sum_b B_{i1}^2 & \sum_b B_{in}^2 \sum_b B_{i2}^2 & \dots & \sum_b B_{in}^2 \sum_b B_{in}^2 \end{bmatrix} \tag{6.8}
\end{aligned}$$

To obtain an equally represented textures in the batch output matrix, I define the distinction score as

$$\begin{aligned}
\text{distinction}(B) &= 1 - \frac{\sum_{i \neq j}^n V_{bij}}{(n-1) \sum_{i=1}^n V_{bii}} \\
&= \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^n (C_{bii} - C_{bjj})^2}{(n-1) \sum_{i=1}^n C_{bii}^2} \\
&= \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^n \left( \sum_{k=1}^b B_{ki} B_{ki} - \sum_{k=1}^b B_{kj} B_{kj} \right)^2}{(n-1) \sum_{i=1}^n \sum_{k=1}^b (B_{ki} B_{ki})^2}
\end{aligned} \tag{6.9}$$

Equation 6.9 suggests that if the diagonal entries of  $C_b$  are equal to each other, the distinction score becomes 0. It also implies that if I want the model to predict equal numbers of textures represented within the mini-batch, the model needs to force the matrix  $C_b$  becoming a scaled identity matrix by equaling the value of the diagonal entries. Thus, for this constraint, the learning model should minimize the distinction score while training.

Following examples illustrate three different cases how distinction score constrains the number of significant textures representing in the matrix  $C_b$ . If there is only one major texture been recognized in the mini-batch, namely only one value arising on the diagonal of  $C_b$ , the distinction score is high.

$$\text{distinction}\left(\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}\right) = 1$$

Nevertheless, the distinction score decreases when more and more distinct textures are represented in a given mini-batch.

$$\text{distinction}\left(\begin{bmatrix} 0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 \\ 0 & 0.8 & 0.2 & 0 & 0 & 0 & 0 & 0 \\ 0.1 & 0.9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.8 & 0.2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.2 & 0 & 0 & 0.8 & 0 \\ 0.8 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 \\ 0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 \\ 0 & 0 & 0.9 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0.9 & 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0.75 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0.8 & 0 & 0 & 0 & 0.2 & 0 & 0 & 0 \end{bmatrix}\right) = 0.65$$

$$\text{distinction}\left(\begin{bmatrix} 0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 \\ 0 & 0.8 & 0.2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.8 & 0.2 \\ 0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0.1 & 0 & 0.9 & 0 & 0 \\ 0 & 0.9 & 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0.6 & 0 & 0 & 0 & 0.4 & 0 & 0 \\ 0 & 0 & 0.8 & 0 & 0.2 & 0 & 0 & 0 \end{bmatrix}\right) = 0.35$$

$$\text{distinction}\left(\begin{bmatrix} 0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 \\ 0 & 0.8 & 0.2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.6 & 0.4 & 0 \\ 0 & 0 & 0 & 0.8 & 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.8 & 0.2 \\ 0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 \\ 0 & 0 & 0 & 0.1 & 0 & 0.9 & 0 & 0 \\ 0 & 0.6 & 0 & 0.4 & 0 & 0 & 0 & 0 \\ 0 & 0.75 & 0 & 0 & 0 & 0.25 & 0 & 0 \\ 0 & 0 & 0.8 & 0 & 0.2 & 0 & 0 & 0 \end{bmatrix}\right) = 0.22$$

At last, the distinction score becomes zero when all textures are represented equally.

$$\text{distinction}\left(\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}\right) = 0$$

Thus, to accomplish the constraint 3, the model is trained to minimize the distinction score over a mini-batch. This constraint is an important factor for the unsupervised labeling model, because it enables the predicted model to recognize different textures. Although it is not able to help the model predict a specific texture label (as is accomplished in the supervised setting), it still informs the meta-task, since the meta-task does not care about any specific texture *label*, only that distinct textures map to different labels.

### 6.2.3.5 Relationships of Distinction and Sparsity to Other Correlation Measures

In the classification setting, there are several common error measures for estimating the proposed partition, such as misclassification rate, negative log-likelihood loss, entropy, cross-entropy, etc.

Here, I explore the relationships between distinction and sparsity score to entropy since they are conceptually similar. Because distinction and sparsity are complementary, I only discuss the relationship between entropy and sparsity, and the relationship between entropy and distinction can be inferred from that.

In information theory, entropy is a measure of predictability of a state. The entropy is defined as

$$H(x) = - \sum_x p(x) \log_2 p(x), \quad (6.10)$$

where  $x$  is a random variable with distribution  $p$ . The maximum entropy is generated when  $x$  is uniformly distributed, and its minimum happens when the distribution of  $x$  has all its mass on one state. Intuitively, the concept of scoring an event based on its probability distribution is very similar to the definition of the sparsity score.

Therefore, I conducted a simple experiment with 100 examples, and each of them has 100 pixels with four different textures. I assigned a label with probability of 0.9 representing the significant texture in this experiment. The first 25 examples have four significant textures, next 25 examples have three, and so on. Following four matrices are the examples illustrating four respective cases.

$$\begin{array}{cc}
 \text{count} = 4 & \text{count} = 3 \\
 P_1 = \begin{bmatrix} \boxed{0.9} & 0.1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \boxed{0.9} & 0.1 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \boxed{0.9} & 0.1 \\ \vdots & \vdots & \vdots & \vdots \\ 0.1 & 0 & 0 & \boxed{0.9} \end{bmatrix} & P_2 = \begin{bmatrix} \boxed{0.9} & 0.1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \boxed{0.9} & 0.1 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \boxed{0.9} & 0.1 \\ \vdots & \vdots & \vdots & \vdots \\ \boxed{0.9} & 0.1 & 0 & 0 \end{bmatrix}
 \end{array}$$

$$\begin{array}{c}
\text{count} = 2 \\
P_3 = \begin{bmatrix} \boxed{0.9} & 0.1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \boxed{0.9} & 0.1 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \boxed{0.9} & 0.1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \boxed{0.9} & 0.1 & 0 \end{bmatrix}
\end{array}
\qquad
\begin{array}{c}
\text{count} = 1 \\
P_4 = \begin{bmatrix} \boxed{0.9} & 0.1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \boxed{0.9} & 0.1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \boxed{0.9} & 0.1 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ \boxed{0.9} & 0.1 & 0 & 0 \end{bmatrix}
\end{array}$$

Since the probabilities of significant textures have been assigned to the same value, entropy undoubtedly has same value for all examples. Nonetheless, the sparsity score can discover different number of significant textures among 100 examples even if the probabilities for significant textures are the same.

I then varied the probabilities for the significant textures, and assign them from 0.75 to 1.0. Figure 6.4 shows the comparison between the entropy and the sparsity score for this experiment. It is obvious that with varied probabilities, entropy still cannot distinguish the number of significant textures that appear in the examples, but the sparsity score can separate them. Although the sparsity score is not perfectly monotonic allowing different numbers of significant textures to result in the same range of the sparsity score, overall the score works as expected.

In conclusion, these two experiments demonstrate that it is hard to accomplish the constraints 2 and 3 by using entropy alone as the regularization term, since it cannot help us identify the counts, which matters in these constraints. Also, uncertainty and sparsity scores share many common terms (dot products of covariance matrix), making it more efficient to compute both.

### 6.2.3.6 Final Loss Function

Given the input  $X$ , the unsupervised labeling model builds a predictive function for regularizing the output matrix based on three terms introduced in the previous sections. The uncertainty and sparsity scores are calculated for every sample and the distinction score is calculated over mini-batches, so I combined the three terms into a single objective by taking the weighted average of

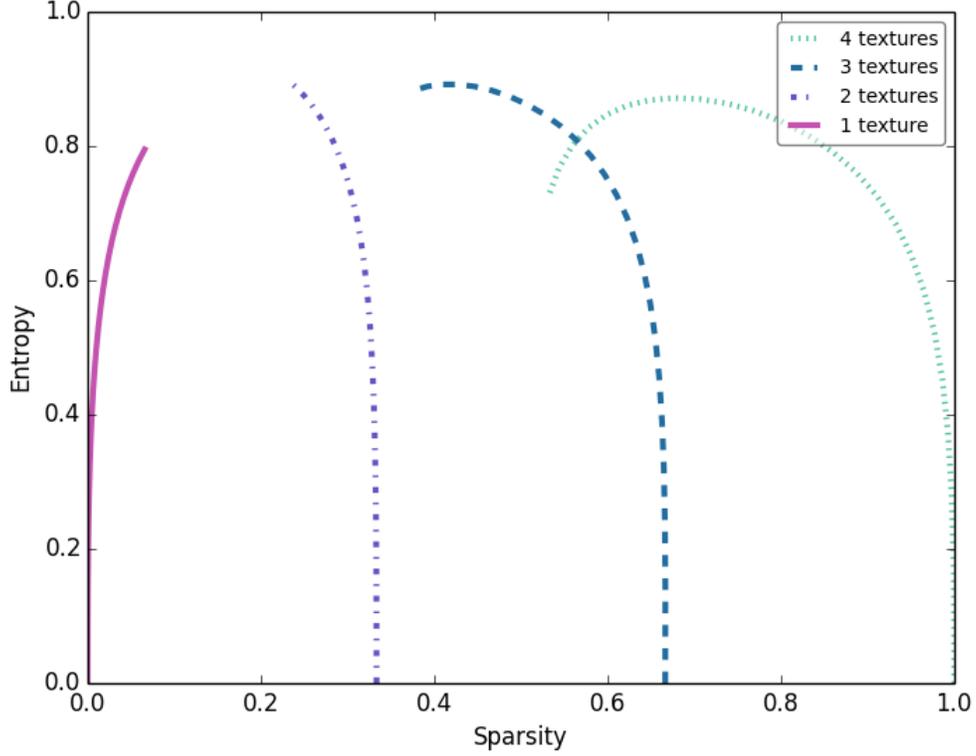


Figure 6.4: Comparison between entropy and the sparsity score with 100 examples. Each example contains 100 pixels within a window, including 4 cases of different number of significant textures appeared within a window. The x-axis represents the sparsity score while the y-axis shows entropy. If the probabilities for all pixels are varied, different number of significant textures can be distinguished by using the sparsity score nevertheless entropy cannot.

uncertainty and sparsity scores over all samples in a mini-batch. Finally, the loss function for the unsupervised labeling model becomes

$$C_{\text{labeling}}(\theta, X) = \text{avg}(c_{\alpha}\text{uncertainty}(P) + c_{\beta}\text{sparsity}(P)) + c_{\gamma}\text{distinction}(B), \quad (6.11)$$

where  $c_{\alpha}$ ,  $c_{\beta}$ ,  $c_{\gamma}$  are the coefficients controlling the balance among these regularizing terms.

For the meta classification task, I train the parameters by minimizing the negative log-likelihood on target  $Z$ :

$$C_{\text{classification}}(\theta, X) = - \sum_{i=0}^{|X|} \log P(Z = z^{(i)} | x^{(i)}, \theta), \quad (6.12)$$

where  $\theta$  is the set of all parameters in the learning model.

Finally, the training objective for the whole model is defined as

$$\underset{\theta}{\text{minimize}} \quad w_h C_{\text{labeling}}(X) + (1 - w_h) C_{\text{classification}}(X), \quad (6.13)$$

where  $w_h \in [0, 1]$  is the guided ratio. If  $w_h = 0$ , the objective is equivalent to  $C_{\text{classification}}$ , and equals to  $C_{\text{labeling}}$  as  $w_h = 1$ . The optimization process can learn different targets by gradually changing the guided ratio. Early in the process the model is trained to learn objective of unsupervised labeling model, and as the process goes on, it starts transiting to train on the target of classification model. Choosing an appropriate guided ratio is essential to my model.

### 6.3 Experimental Results

I evaluated the proposed approach on Brodatz texture dataset described in Section 5.3. The setup for the experiments is identical to the one that used in Section 5.3.2. In this section, I show the labeling results with different weighting coefficients for the regularizers. Also, the comparison between the sparsity and distinction scores with and without supervision is discussed in this section.

#### 6.3.1 Evaluation on Regularization Terms

Experiments were performed using two different settings to demonstrate the effect of the regularization terms on the output distribution. Searching an appropriate coefficient set is easier if all coefficients sum to 1, decreasing the dimensions of coefficient space down to two. Moreover, this simplified the experiments by controlling the coefficients of the uncertainty and sparsity scores and experimenting on the coefficient of the distinction score  $c_\gamma$ , because I assume that the ability to distinguish different textures, distinction, is the most important constraint of the three.

I emphasized the importance of mini-batch distinguish by giving  $c_\gamma$  the highest weight ( $c_\alpha = 1, c_\beta = 1, c_\gamma = 10$ ), driving the textures representing within a mini-batch more uniformly. Figure 6.5 visualizes the cumulative probability distribution over all samples within a mini-batch with higher coefficient for the distinction score. The flat distribution shown in Figure 6.5 corroborates

that exploiting the distinction regularization term can help the model recognize more textures shown within a mini-batch.

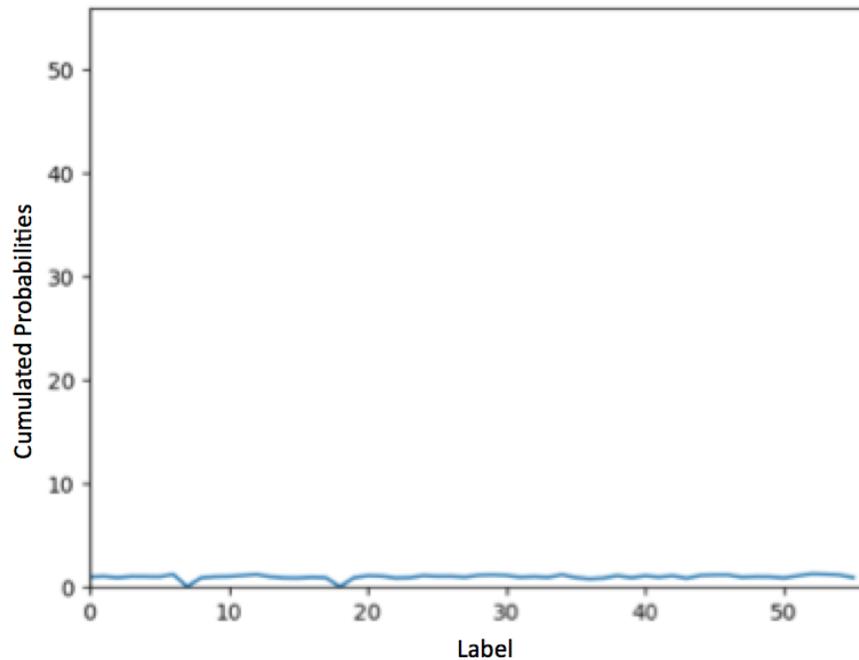


Figure 6.5: A cumulative probability distribution over all samples within a mini-batch with high coefficient for the distinction score.

I then examined the cumulative probability distribution over all pixels within a window to evaluate the other two scores. Figure 6.6 shows the samples of same texture and different textures for foreground and background in left and right, respectively. With high  $c_\gamma$ , it is surprised that the model has difficulty in reducing the number of texture labels within a window, which is restrained by the sparsity score. The sparsity score is computed on a single image basis, while distinction is calculated over all images within a mini-batch. The model is supposed to be able to minimize both of them at the same time.

In a second study, lowered  $c_\gamma$  in an effort to allow the sparsity score ( $c_\beta$ ) to dominate the loss function ( $c_\alpha = 1, c_\beta = 20, c_\gamma = 7$ ). The number of texture labels within a window decreases, and in Figure 6.7 the probability distribution shows one peak for same texture and two peaks for different, which is perfectly satisfied to the constraint 2. However, the cumulative probability distribution

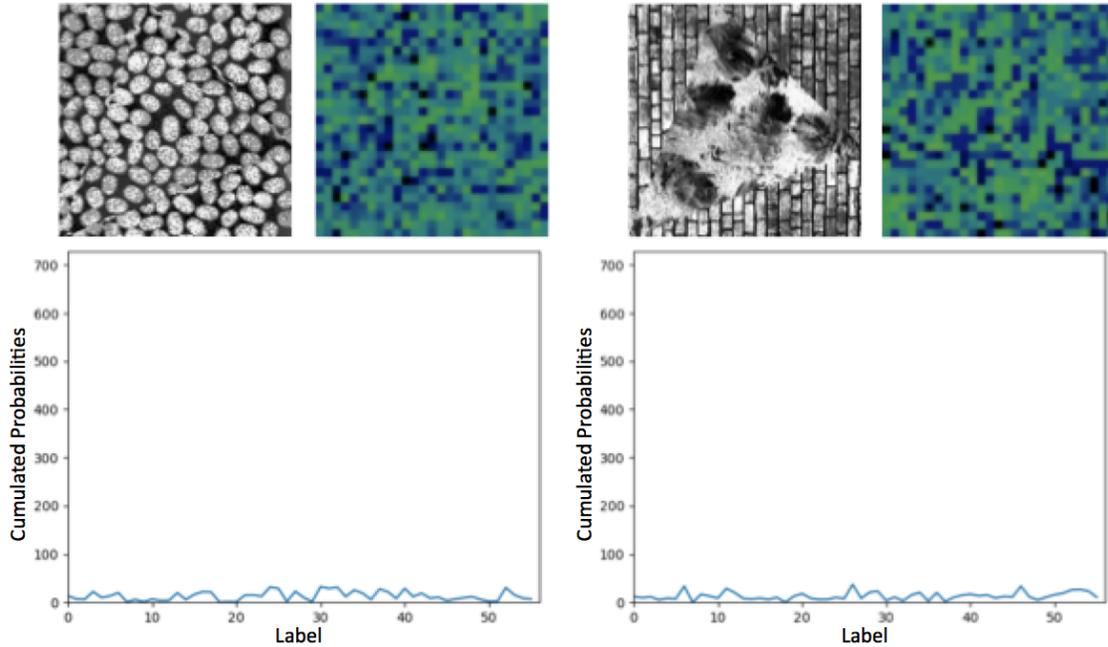


Figure 6.6: Examples of the same and different textures for foreground and background with high coefficient on the distinction score. When the coefficient of the distinction score sets high, the cumulative probability distribution shows that most of the textures have been recognized within a mini-batch and the false-color images also reflect the effects.

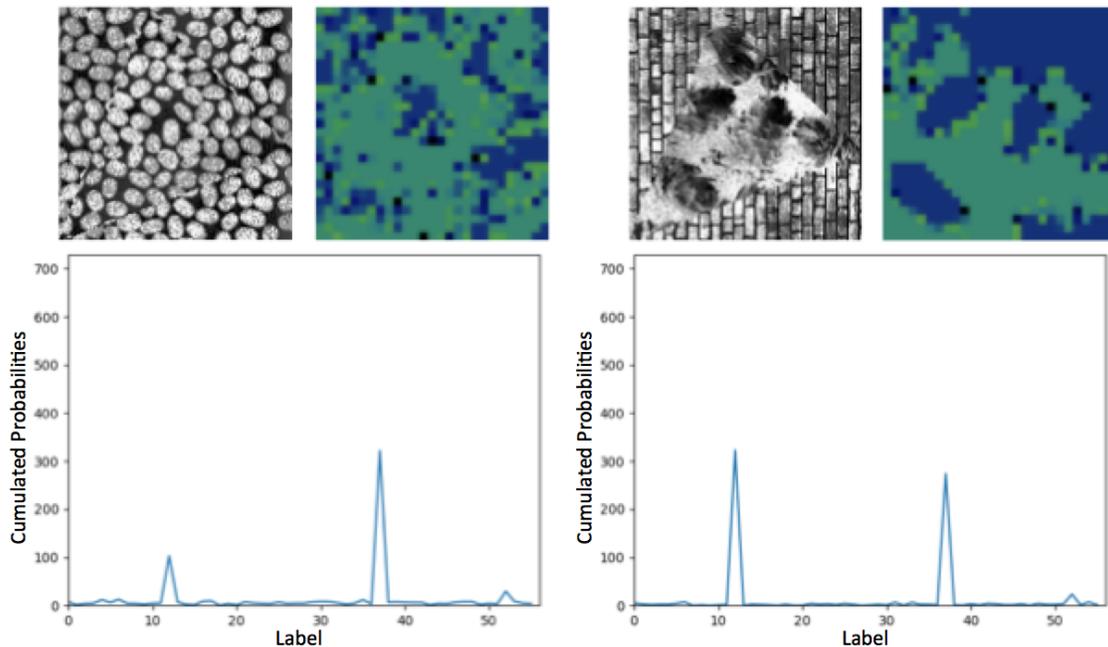


Figure 6.7: Examples of same and different textures for foreground and background with high coefficient on the sparsity score. From the cumulative probability distribution, the number of the textures been recognized within a mini-batch decreases to one significant peak for the same texture and two significant peaks for different, when the sparsity score sets high. There are also fewer colors dominating in the false-color images.

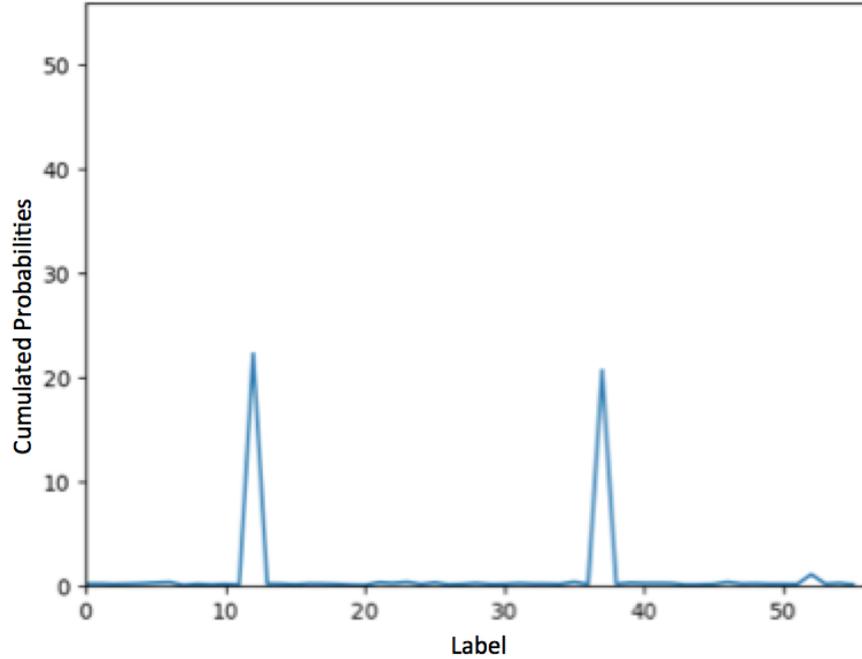


Figure 6.8: A cumulative probability distribution over all samples within a mini-batch with lower coefficient for the distinction score.

over all samples within a mini-batch forms two peaks as well, meaning that the constraint 3 is not achieved and the model fails to minimize the distinction score.

The results of these two contrastive examples indicate that the model is very sensitive to the weighting assigned to sparsity and distinction scores. This motivated me to examine further the relationship between the sparsity and distinction scores as discussed in the following section.

### 6.3.2 Comparison Between Sparsity and Distinction

Considering the effect of training the sparsity and distinction scores with different weighting coefficients, the results is illustrated in Figure 6.9. From the observation, these two scores tend to show reciprocal relationships between each other. It shows that when  $c_\gamma$  sets high, it gets lower distinction score but high sparsity score, meaning that many textures not only show within a mini-batch but also in every window. On the other hand, when  $c_\beta$  becomes higher, the sparsity score starts dropping with increasing distinction score. In this case, the model is able to control the number of textures shown in a window but unfortunately also only few textures are recognized within a

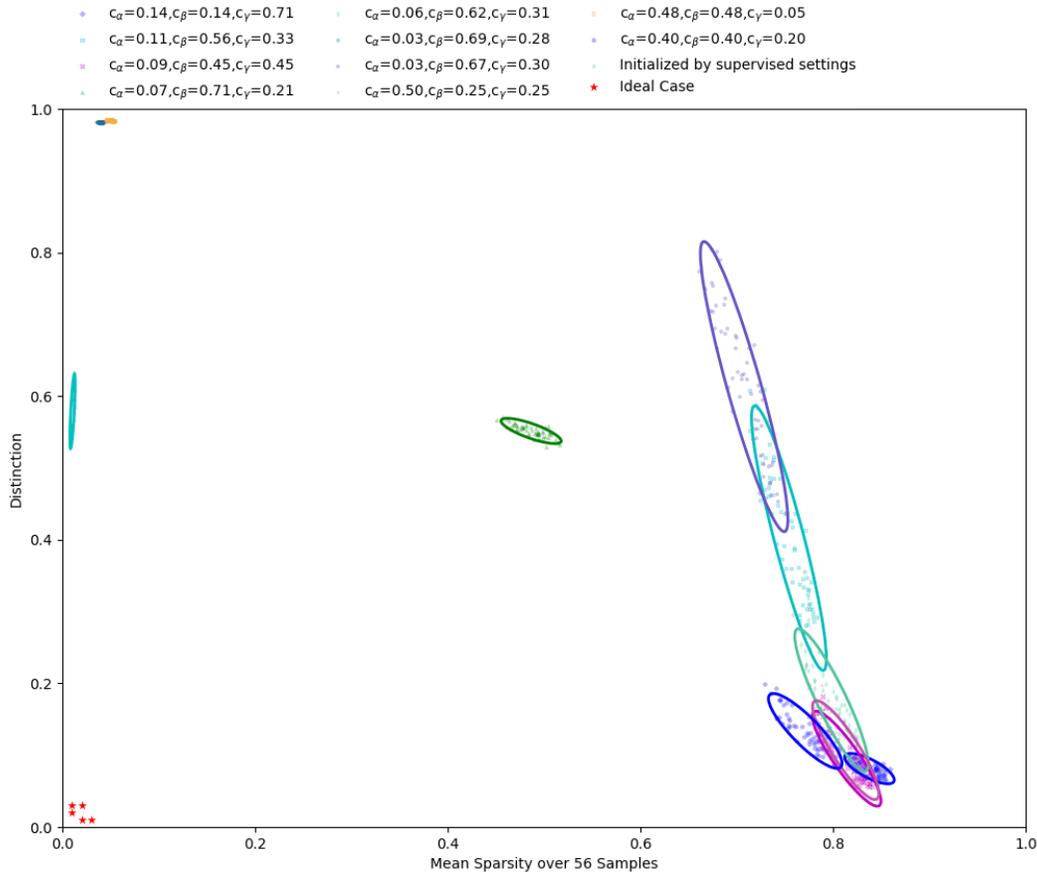


Figure 6.9: Comparison between the distinction and sparsity score with different coefficient settings. Note that the x-axis is the mean sparsity score over all samples within a mini-batch.

mini-batch. It implies that when the distinction score is low, showing uniformly represented textures within a mini-batch, the model does not learn to distribute different textures to every single sample; instead, all textures appear in every sample. The ideal case is to have lower values on both sparsity and distinction scores, which is shown in red in Figure 6.9, saying that all textures are evenly represented within a mini-batch and every particular texture is evenly distributed to each sample within the mini-patch, leading to few textures showing in every sample. The results initialized by supervised settings are shown in the teal cluster in the left side of the Figure 6.9, with low sparsity score and middle distinction score.

The objective of our unsupervised pixel labeling model is to approximate the performance of the supervised setting. The observation raises the following questions that is there a way that can

lead the model to learn from the extreme case to the supervised one? The results suggest that the three regularization terms are necessary but not sufficient for solving the meta-task. If I predict the model initialized by a supervised pretraining using a small subset of samples with known labels, can it lead the model to achieve as good as our supervised model? Moreover, the model trained for the experiments is based on the assumption that a fixed optimal coefficient set exists across different guided ratios over training time. However, might it be possible that the optimal solution shifts in the coefficient space across different guided ratios during training time, rather than sticks to a fixed point? To discover a better solution, these are potential directions for future exploration.

## **6.4 Conclusion and Discussion**

In this chapter, I extended the labeling model to learn in an unsupervised fashion. Inferred by the predicted output of the labeling model, I introduced an unsupervised learning objective considering the correlation relationships and distinction patterns among the samples. Three regularization terms, uncertainty, sparsity, and distinction, are designed to refine the label assignments by iteratively constructing an expected output distribution during the unsupervised labeling training. No additional labeled dataset is required for training. The labeling predictions are updated only based on the regularizers.

Experiments demonstrate that the constraints can be satisfied by minimizing the defined regularization terms. However, setting weights for the three constraint terms is challenging. The model is trained on the basis of an assumption that the optimal coefficient set for three terms is fixed across different guided ratios over training time. But there is a possibility that the optimal coefficient set is changed over different guided ratios. Also, it might be possible that the three regularization terms are necessary but not sufficient to solve the problem. In this case, semi-supervised training with initialized learning parameters might be able to tackle the problem. With a small subset of samples with known labels, semi-supervised approach should be able to guide the model to the right direction. The pixel labeling model is demonstrated to provide semantic hints to the subsequent tasks. Achieving useful pixel labels in an unsupervised setting is still a topic for future research.

## CHAPTER 7: CONCLUSIONS AND FUTURE WORK

In this dissertation, I presented two deep learning architectures to assist in histology image analysis. Throughout the work, the guiding principle was to capture important feature representations that provide valuable and useful hints for a subsequent classification task. I have also demonstrated how the proposed deep learning approaches generalize to a variety of texture-oriented problems. In the sections below, I summarize my results and propose possible avenues for future research.

### 7.1 Extensions to Combine Features Across Scales

In Chapter 3, I proposed a learning architecture which can extract appropriate features, visualize the features learned by the model, and classify the images based on the features learned. The novel deep learning architecture adds an intermediate layer, called a hyperlayer, to capture features learned from different convolutional layers. With a hyperlayer, the learning network is able to combine representations from different levels of abstraction and scales of an image, and thus provide a more complete description for the classification process.

The proposed architecture was evaluated on a histology image dataset. The results show that the proposed approach outperforms the canonical methods and has captured and learned important features and details at lower layers which are critical in the dataset. The improvements due to the hyperlayer suggest that the introduction of this intermediate layer to the learning model makes the classification task easier and enables it to learn faster.

Although the proposed architecture improves classification accuracy, it is limited in its ability to scale. One reason is that the hyperlayer relies on learned representations from all of the convolutional layers below it. The output for all convolutional layers in a very deep architecture might be overwhelming, leading to long training times and large memory requirements. One simple solution

might combine the representations from *selected* convolutional layers, instead of taking all of them, to the hyperlayer. This solution not only addresses the scalability issue but might also filter out insignificant representations.

In addition, since the proposed architecture utilizes autoencoders for intermediate levels of unsupervised feature learning, the proposed approach inherits the limitations of autoencoders. One of the limitations is that the features learned from the autoencoder might not be meaningful semantics or useful representations for the desired classification task. The autoencoder is optimized by minimizing the difference between the input and the reconstructed output rather than its sensitivity to a given task.

An alternate approach, which more closely mimics a traditional histology analysis pipeline, attempts to label the tissues and cell types prior to overall classifications. One solution might capture the cell-type representations as a supervised objectives, rather than expecting the autoencoder to capture them automatically. If the image representations are forced to a finite set of labels of meaningful cell/tissue types, it also makes the dataset more consistent. Looking at a region of labels would provide an overall picture of the state of a given tissue, and aid the pathologists to determine whether the tissue is normal or abnormal. A more comprehensive introduction of combining pixel labeling to capture semantic information to the learning process is detailed in Chapter 4.

## **7.2 Extension to Supervised Pixel/Region Labeling**

In Chapter 4, I extended the model by introducing a specific intermediate learning objective, which is a per pixel labeling to provide semantic hints to subsequent classification layers, in the spirit of curriculum learning. Instead of directly classifying the images, the model first learns a label for the center pixel given the context of its neighbors. The image classifier then uses these labels to classify the images. The main contributions of this work are a novel deep learning architecture for image classification that propagates specific domain knowledge from the pixel level to the image level.

My experiments suggest that image classification does not require an extremely accurate pixel labeling. Furthermore, the pixel labeling layer can be regarded as a vector of label probabilities, that are provided as hints for subsequent image classification. Extensive experiments show that my architecture outperforms other traditional deep learning approaches. The experiments demonstrated that the pixel labeling not only can provide robust and meaningful semantic representations to the subsequent task but also lead to more accurate and efficient learning.

However, one might think that an image labeling might be too specific to histology datasets. The improved performance is achieved by the efforts of the people in the domain coming up with well-designed set of labels, allowing the pixel labeling model learn more easily. An extension to explore whether a pixel labeling layer can generalize to different learning problems beyond histology images was introduced in Chapter 5.

In Chapter 5, the proposed supervised labeling model was applied to two toy problems, a Pentomino problem and a texture-oriented problem, to demonstrate the general utility of my labeling model. My work has demonstrated that the labeling approach generalizes to other vision tasks, and is robust to noise. In addition, this work demonstrates that the labeling model is beneficial for classifying texture datasets. It is able to capture low-level information in particular and converts it to valuable hints.

Although the pixel labeling proves to be generalizable, it has a disadvantage that it relies on a set of well-designed, domain specific, labels. Currently, the selection of the meaningful labels and the dataset for the pixel labeling layer are manually selected by the domain experts. Since the labels are domain specific, not task specific, the predefined labels need to be redesigned if the classification problem changes. For example, if we try to classify the traffic scene, the predefined labels could be modified to include vehicles, buildings, sky, road, sidewalks, pedestrians, etc.

Granularity and the number of the predefined labels are also important factors when selecting labels. These two factors are reciprocal; if the granularity is larger, the hierarchy of the label design would be shallower, and thus the number of the predefined labels would be fewer. In contrast, when the granularity is small, the hierarchy of labels might become big and deep, leading to a large

amount of labels. Some evidence based on limited testing shows that it is not necessary to set the granularity very small. Unless the features for individual label are specific and unique, adding more labels might result in worse prediction performance.

In addition to the label selection, one might also argue that the proposed pixel labeling model is less appealing in due its need for an extra dataset for training the labeling model. Although my proposed pixel labeling approach did not require dense labels it did require expert curation, which involved the hand-selection of many thousands of representative label examples. The limitations of this labeling model might be solved by making the pixel labeling model learn in an unsupervised setting. If the model can learn the labels by using statistical information of the data, and let the data itself suggest the labels for the intermediate layer, the process of designing a set of labels and collecting an extra training dataset for the labeling model can be eliminated. An unsupervised pixel labeling model, which is introduced in Chapter 6, is then proposed to address the issues.

In Chapter 6, the labeling model is further extended to an unsupervised setting. Inferred by the predicted output of the labeling model, three regularization terms, uncertainty, sparsity, and distinction, are designed to refine the label assignments by gradually shaping the expected output distribution during the unsupervised labeling training. Without additional labeled dataset involved in the learning process, the experiment results demonstrate that the constraints can be satisfied by minimizing the defined regularization terms.

However, selecting appropriate weights for all three terms in order to simultaneously satisfy all of the desired constraints of the output distribution is challenging. Based on limited testing, it is hard to conclude that the optimal coefficient set lies at a fixed point in the parameter space over the training time. It could be a possibility that the optimal coefficient set must be changed over time to achieve an effective ratio. One future direction for the work would be to explore an efficient way for searching an optimal coefficient set for the three regularizers.

Another future direction for the work is to apply the unsupervised settings to the histology images dataset. Currently, the unsupervised labeling model is a proof of concept realized on a toy

problem. More experimentation is required to demonstrate that applying an unsupervised labeling model can improve the classification in real-world images.

Another challenge might emerge when the unsupervised labeling model I proposed is applied to a histology dataset. A strong assumption is made in my toy problem with regard to the distinction score. It assumes that all labels are present uniformly within a randomized mini-batch, which might be not easy to satisfy. A tubule image might compose several labels, and even more, the same stages of the tubule might include different amount of labels. To relax this assumption on the distinction score, one solution might be to collect the statistics of the numbers of labels that appear in different stages and arrange the training data within a mini-batch based on the statistics. Although it cannot guarantee the labels are uniformly represented in the mini-batch, the ratio among respective labels can be calculated, and the distinction score can be tuned correspondingly to achieve the ratio.

Regarding to the applications of pixel labeling, other than the classification task, pixel labeling can facilitate quantitative assessment of variation for the number of tubule cross sections and the number of tubules with defects. Pixel labeling can be used to infer the fraction of the total cross section area for cell types of interest, providing valuable information for the overall histology image analysis. However, the quantitative assessment, unlike the classification task in our pipeline, requires higher pixel labeling accuracy. One possible way to boost the pixel labeling accuracy is to incorporate feedback from domain experts and continue to refine the predictions or reshape the probability distribution of labels iteratively based on the immediate feedback. This is contrast to the common learning paradigm in which the model is learned only from the static training dataset. One of the major problems with today's machine learning methods is that they are too dependent upon their training data. In traditional machine learning pipeline, the model is learned only once, and thus is unable to adapt to either technical variants, such as changes in sample preparation, or systemic changes, such as an evolution of scoring criteria. An online learning framework for pixel labeling might be a research area worthwhile to explore in the future.

### 7.3 Conclusion

In conclusion, I set out to improve the classification in histology images in deep learning framework by introducing intermediate learning objectives. I have shown two different types of intermediate learning objectives, which dramatically improve over generic deep learning models used in computer vision. One is combined autoencoder and hyperlayer to do completely unsupervised learning of things in different scales. The advantage of this method is that it does not need additional training dataset. The problem of the hyperlayer is that there is no guarantee that the features learned to reconstruct the images are semantically meaningful.

Another solution which uses semantics as hints, called pixel labeling, is then proposed. I have demonstrated how to incorporate this common technique that people used in histology image, to deep learning framework. Illustrated by doing that, I can get much improved performance. The pixel labeling shows to be a very powerful intermediate objective handling hard to learn problem and is demonstrated that it can be generalized to more than just histology images.

The downside of that is it requires a new training dataset, thus increasing the amount of upfront work needed to be done. Therefore, I still would like to continue to develop unsupervised approaches like my hyperlayer approach. Doing that, it must be possible to capture important features of pixel labeling without explicitly doing so, and have a system that will find the pixel labels like features as intuitive as the autoencoder with unsupervised setting. In an ideal world, there would be a system that basically takes the input images and clusters the pixels to several different classes.

## BIBLIOGRAPHY

- Amin, S., Filippas, J., Naguib, R., and Bennett, M. K. (2003). A parallel system for performing colonic tissue classification by means of a genetic algorithm. In *European Parallel Virtual Machine/Message Passing Interface Users Group Meeting*, pages 560–564. Springer.
- Arif, M. and Rajpoot, N. (2007). Classification of potential nuclei in prostate histology images using shape manifold learning. In *International Conference on Machine Vision*, pages 113–118. IEEE.
- Bengio, Y. (2012). Deep learning of representations for unsupervised and transfer learning. In *International Conference on Machine Learning*, pages 17–36.
- Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.
- Bengio, Y. et al. (2009a). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.
- Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, pages 153–160.
- Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009b). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48.
- Berg-Kirkpatrick, T., Bouchard-Côté, A., DeNero, J., and Klein, D. (2010). Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590. Association for Computational Linguistics.
- Boucheron, L. E. (2008). *Object-and spatial-level quantitative analysis of multispectral histopathology images for detection and characterization of cancer*. University of California at Santa Barbara.
- Boucheron, L. E., Bi, Z., Harvey, N. R., Manjunath, B., and Rimm, D. L. (2007). Utility of multispectral imaging for nuclear classification of routine clinical histopathology imagery. *BMC Cell Biology*, 8(1):S8.
- Bovik, A. C., Clark, M., and Geisler, W. S. (1990). Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):55–73.
- Brodatz, P. (1966). *Textures: a photographic album for artists and designers*. Dover Publications, Inc, 1 edition.
- Can, A., Bello, M., Tao, X., Gerdes, M., Sood, A., Montalto, M., and Ginty, F. (2008). Techniques for cellular quantitation of cancer biomarkers. *Microscopic Image Analysis for Life Science Applications*.

- Chang, H., Nayak, N., Spellman, P. T., and Parvin, B. (2013). Characterization of tissue histopathology via predictive sparse decomposition and spatial pyramid matching. In *International Conference on Medical Image Computing and Computer-assisted Intervention*.
- Chen, C.-H., Huang, W.-T., Tan, T.-H., Chang, C.-C., and Chang, Y.-J. (2015). Using k-nearest neighbor classification to diagnose abnormal lung sounds. *Sensors*, 15(6):13132–13158.
- Chen, X. and Gupta, A. (2015). Weakly supervised learning of convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1431–1439.
- Cireşan, D., Giusti, A., Gambardella, L. M., and Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in Neural Information Processing Systems*, pages 2843–2851.
- Cireşan, D. C., Giusti, A., Gambardella, L. M., and Schmidhuber, J. (2013). Mitosis detection in breast cancer histology images with deep neural networks. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pages 411–418. Springer.
- Collaborative Cross Consortium et al. (2012). The genome architecture of the collaborative cross mouse genetic reference population. *Genetics*, 190(2):389–401.
- Cosatto, E., Miller, M., Graf, H. P., and Meyer, J. S. (2008). Grading nuclear pleomorphism on histological micrographs. In *19th International Conference on Pattern Recognition*, pages 1–4. IEEE.
- Cruz-Roa, A., Basavanthally, A., González, F., Gilmore, H., Feldman, M., Ganesan, S., Shih, N., Tomaszewski, J., and Madabhushi, A. (2014). Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks. In *SPIE Medical Imaging*, volume 9041, pages 904103–904103. International Society for Optics and Photonics.
- Cruz-Roa, A. A., Ovalle, J. E. A., Madabhushi, A., and Osorio, F. A. G. (2013). A deep learning architecture for image representation, visual interpretability and automated basal-cell carcinoma cancer detection. In *International Conference on Medical Image Computing and Computer-assisted Intervention*, pages 403–410. Springer.
- Dalle, J.-R., Li, H., Huang, C.-H., Leow, W. K., Racoceanu, D., and Putti, T. C. (2009). Nuclear pleomorphism scoring by selective cell nuclei detection. In *WACV*.
- Díaz-Pernas, F., Antón-Rodríguez, M., Díez-Higuera, J., Martínez-Zarzuela, M., González-Ortega, D., and Boto-Giralda, D. (2009). Texture classification of the entire brodatz database through an orientational-invariant neural architecture. In *International Work-Conference on the Interplay Between Natural and Artificial Computation*, pages 294–303. Springer.
- Dizaji, K. G., Herandi, A., Deng, C., Cai, W., and Huang, H. (2017). Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *2017 IEEE International Conference on Computer Vision*, pages 5747–5756. IEEE.
- Dong, Q., Gong, S., and Zhu, X. (2016). Multi-task curriculum transfer deep learning of clothing attributes. *arXiv preprint arXiv:1610.03670*.

- Doyle, S., Rodriguez, C., Madabhushi, A., Tomaszewski, J., and Feldman, M. (2006). Detecting prostatic adenocarcinoma from digitized histology using a multi-scale hierarchical classification approach. In *28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4759–4762. IEEE.
- Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.
- Ertosun, M. G. and Rubin, D. L. (2015). Automated grading of gliomas using deep learning in digital pathology images: A modular approach with ensemble of convolutional neural networks. In *AMIA Annual Symposium Proceedings*, volume 2015, page 1899. American Medical Informatics Association.
- Everingham, M., Van Gool, L., Williams, C., Winn, J., and Zisserman, A. (2011). The pascal visual object classes challenge 2012 (voc2012) results (2012). In URL <http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html>.
- Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2013). Learning hierarchical features for scene labeling. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929.
- Fernandez, D. C., Bhargava, R., Hewitt, S. M., and Levin, I. W. (2005). Infrared spectroscopic imaging for histopathologic recognition. *Nature Biotechnology*, 23(4):469–474.
- Gil, J., Wu, H., and Wang, B. Y. (2002). Image analysis and morphometry in the diagnosis of breast cancer. *Microscopy Research and Technique*, 59(2):109–118.
- Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.
- Gong, C., Tao, D., Maybank, S. J., Liu, W., Kang, G., and Yang, J. (2016). Multi-modal curriculum learning for semi-supervised image classification. *IEEE Transactions on Image Processing*, 25(7):3249–3260.
- Gülçehre, Ç. and Bengio, Y. (2016). Knowledge matters: Importance of prior information for optimization. *Journal of Machine Learning Research*, 17(8):1–32.
- Gurcan, M. N., Boucheron, L. E., Can, A., Madabhushi, A., Rajpoot, N. M., and Yener, B. (2009). Histopathological image analysis: A review. *Biomedical Engineering*, 2:147–171.
- Hafemann, L. G., Oliveira, L. S., and Cavalin, P. (2014). Forest species recognition using deep convolutional neural networks. In *International Conference on Pattern Recognition*, pages 1103–1107. IEEE.
- Haralick, R. M., Shanmugam, K., et al. (1973). Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, (6):610–621.

- Hattel, A., Monga, V., Srinivas, U., Gillespie, J., Brooks, J., Fisher, J., and Jayarao, B. (2013). Development and evaluation of an automated histology classification system for veterinary pathology. *Journal of Veterinary Diagnostic Investigation*, 25(6):765–769.
- He, D.-C. and Wang, L. (1990). Texture unit, texture spectrum, and texture analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 28(4):509–512.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.
- He, L., Long, L. R., Antani, S., and Thoma, G. (2010). Computer assisted diagnosis in histopathology. *Sequence and Genome Analysis: Methods and Applications*, 510:271–287.
- Heaton, J. (2017). Ian goodfellow, yoshua bengio, and aaron courville: Deep learning.
- Hinton, G. E. (1990). Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, 46(1-2):47–75.
- Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.
- Huang, X., Zhang, L., and Wang, L. (2009). Evaluation of morphological texture features for mangrove forest mapping and species discrimination using multispectral ikonos imagery. *IEEE Geoscience and Remote Sensing Letters*, 6(3):393–397.
- Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456.
- Japkowicz, N., Hanson, S. J., and Gluck, M. A. (2000). Nonlinear autoassociation is not equivalent to pca. *Neural Computation*, 12(3):531–545.
- Jiang, L., Meng, D., Yu, S.-I., Lan, Z., Shan, S., and Hauptmann, A. (2014). Self-paced learning with diversity. In *Advances in Neural Information Processing Systems*, pages 2078–2086.
- Junqueira, L. C. and Carneiro, J. (2005). *Basic histology text and atlas*. London: McGraw Hill, 2005.
- Kao, C.-Y., Madduri, M., and McMillan, L. (2017). A deep learning architecture for histology image classification with curriculum learning. In *European Congress on Computational Methods in Applied Sciences and Engineering*, pages 1102–1111. Springer.
- Kao, C.-Y. and McMillan, L. (2017). A novel deep learning architecture for testis histology image classification. *arXiv preprint arXiv:1707.05809*.
- Karhunen, J. and Joutsensalo, J. (1995). Generalizations of principal component analysis, optimization problems, and neural networks. *Neural Networks*, 8(4):549–562.

- Karpathy, A. and Van De Panne, M. (2012). Curriculum learning for motor skills. In *Canadian Conference on Artificial Intelligence*, pages 325–330. Springer.
- Kather, J. N., Weis, C.-A., Bianconi, F., Melchers, S. M., Schad, L. R., Gaiser, T., Marx, A., and Zöllner, F. G. (2016). Multi-class texture analysis in colorectal cancer histology. *Scientific reports*, 6:27988.
- Kayser, G., Riede, U., Werner, M., Hufnagl, P., and Kayser, K. (2002). Towards an automated morphological classification of histological images of common lung carcinomas. *Electronic Journal of Pathology and Histology*, 8:022–03.
- Kilinc, O. and Uysal, I. (2017). Gar: An efficient and scalable graph-based activity regularization for semi-supervised learning. *arXiv preprint arXiv:1705.07219*.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105.
- Kumar, M. P., Packer, B., and Koller, D. (2010a). Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197.
- Kumar, M. P., Packer, B., and Koller, D. (2010b). Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197.
- Laine, A. and Fan, J. (1993). Texture classification by wavelet packet signatures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1186–1191.
- Lannin, T. B., Thege, F. I., and Kirby, B. J. (2016). Comparison and optimization of machine learning methods for automated classification of circulating tumor cells. *Cytometry Part A*, 89(10):922–931.
- Lanning, L. L., Creasy, D. M., Chapin, R. E., Mann, P. C., Barlow, N. J., Regan, K. S., and Goodman, D. G. (2002). Recommended approaches for the evaluation of testicular and epididymal toxicity. *Toxicologic Pathology*, 30(4):507–520.
- LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551.
- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- Lee, Y. J. and Grauman, K. (2011). Learning the easy things first: Self-paced visual category discovery. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1721–1728. IEEE.
- Li, M., Zhang, T., Chen, Y., and Smola, A. J. (2014). Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 661–670. ACM.

- Litjens, G., Sánchez, C. I., Timofeeva, N., Hermsen, M., Nagtegaal, I., Kovacs, I., Hulsbergen-Van De Kaa, C., Bult, P., Van Ginneken, B., and Van Der Laak, J. (2016). Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis. *Scientific reports*, 6:26286.
- Liu, L. and Fieguth, P. (2012). Texture classification from random features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):574–586.
- Malon, C. D. and Cosatto, E. (2013). Classification of mitotic figures with convolutional neural networks and seeded blob features. *Journal of Pathology Informatics*, 4.
- Masci, J., Meier, U., Cireşan, D., and Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. *Artificial Neural Networks and Machine Learning*, pages 52–59.
- Massar, M. L., Bhagavatula, R., Ozolek, J. A., Castro, C. A., Fickus, M., and Kovacevic, J. (2011). A domain-knowledge-inspired mathematical framework for the description and classification of h&e stained histopathology images. In *Proceedings of SPIE*, pages 81380U–81380U.
- McCann, M. T. (2015). Tools for automated histology image analysis.
- McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.
- Meistrich, M. L. and Hess, R. A. (2013). Assessment of spermatogenesis through staging of seminiferous tubules. *Spermatogenesis: Methods and Protocols*, pages 299–307.
- Mills, S. E. (2012). *Histology for pathologists*. Lippincott Williams & Wilkins, 4 edition.
- Nateghi, R., Danyali, H., SadeghHelfroush, M., and Pour, F. P. (2014). Automatic detection of mitosis cell in breast cancer histopathology images using genetic algorithm. In *Iranian Conference on Biomedical Engineering*, pages 1–6. IEEE.
- Neelakantan, A., Vilnis, L., Le, Q. V., Sutskever, I., Kaiser, L., Kurach, K., and Martens, J. (2015). Adding gradient noise improves learning for very deep networks. *arXiv preprint arXiv:1511.06807*.
- Nguyen, K., Jain, A. K., and Sabata, B. (2011). Prostate cancer detection: Fusion of cytological and textural features. *Journal of pathology informatics*, 2.
- Oakberg, E. F. (1956). A description of spermiogenesis in the mouse and its use in analysis of the cycle of the seminiferous epithelium and germ cell renewal. *Developmental Dynamics*, 99(3):391–413.
- Odet, F., Pan, W., Bell, T. A., Goodson, S. G., Stevans, A. M., Yun, Z., Aylor, D. L., Kao, C.-Y., McMillan, L., de Villena, F. P.-M., et al. (2015). The founder strains of the collaborative cross express a complex combination of advantageous and deleterious traits for male reproduction. *G3: Genes, Genomes, Genetics*, 5(12):2671–2683.

- Pentina, A., Sharmanska, V., and Lampert, C. H. (2015). Curriculum learning of multiple tasks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5492–5500.
- Phillips, K. G., Kolatkar, A., Rees, K. J., Rigg, R., Marrinucci, D., Luttgen, M., Bethel, K., Kuhn, P., and McCarty, O. J. (2012a). Quantification of cellular volume and sub-cellular density fluctuations: comparison of normal peripheral blood cells and circulating tumor cells identified in a breast cancer patient. *Frontiers in Oncology*, 2.
- Phillips, K. G., Velasco, C. R., Li, J., Kolatkar, A., Luttgen, M., Bethel, K., Duggan, B., Kuhn, P., and McCarty, O. J. (2012b). Optical quantification of cellular mass, volume, and density of circulating tumor cells identified in an ovarian cancer patient. *Frontiers in Oncology*, 2.
- Poultney, C., Chopra, S., Cun, Y. L., et al. (2007). Efficient learning of sparse representations with an energy-based model. In *Advances in Neural Information Processing Systems*, pages 1137–1144.
- Riordan, D. P., Varma, S., West, R. B., and Brown, P. O. (2015). Automated analysis and classification of histological tissue features by multi-dimensional microscopic molecular profiling. *PLoS one*, 10(7):e0128975.
- Rosenblatt, F. (1962). Principles of neurodynamics.
- Rumelhart, D. E. and McClelland, J. L. (1986). Parallel distributed processing: Explorations in the microstructure of cognition: Foundations (parallel distributed processing).
- Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244.
- Shorter, J. R., Odet, F., Aylor, D. L., Pan, W., Kao, C.-Y., Fu, C.-P., Morgan, A. P., Greenstein, S., Bell, T. A., Stevens, A. M., et al. (2017). Male infertility is responsible for nearly half of the extinction observed in the mouse collaborative cross. *Genetics*, 206(2):557–572.
- Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- Sims, A., Bennett, M., and Murray, A. (2003). Image analysis can be used to detect spatial changes in the histopathology of pancreatic tumours. *Physics in Medicine and Biology*, 48(13):N183.
- Sirinukunwattana, K., Raza, S. E. A., Tsang, Y.-W., Snead, D. R., Cree, I. A., and Rajpoot, N. M. (2016). Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images. *IEEE Transactions on Medical Imaging*, 35(5):1196–1206.
- Svensson, C.-M., Hübler, R., and Figge, M. T. (2015). Automated classification of circulating tumor cells and the impact of interobserver variability on classifier training and performance. *Journal of Immunology Research*, 2015.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

- Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408.
- Voisin, A., Krylov, V. A., Moser, G., Serpico, S. B., and Zerubia, J. (2013). Classification of very high resolution sar images of urban areas using copulas and texture in a hierarchical markov random field model. *IEEE Geoscience and Remote Sensing Letters*, 10(1):96–100.
- Wang, H., Cruz-Roa, A., Basavanhally, A., Gilmore, H., Shih, N., Feldman, M., Tomaszewski, J., Gonzalez, F., and Madabhushi, A. (2014a). Cascaded ensemble of convolutional neural networks and handcrafted features for mitosis detection. In *SPIE Medical Imaging*, volume 9041, pages 90410B–90410B. International Society for Optics and Photonics.
- Wang, H., Cruz-Roa, A., Basavanhally, A., Gilmore, H., Shih, N., Feldman, M., Tomaszewski, J., Gonzalez, F., and Madabhushi, A. (2014b). Mitosis detection in breast cancer pathology images by combining handcrafted and convolutional neural network features. *Journal of Medical Imaging*, 1(3):034003–034003.
- Wei, N., You, J., Friehs, K., Flaschel, E., and Nattkemper, T. W. (2007). An in situ probe for on-line monitoring of cell density and viability on the basis of dark field microscopy in conjunction with image processing and supervised machine learning. *Biotechnology and Bioengineering*, 97(6):1489–1500.
- Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits. Technical report, STANFORD UNIV CA STANFORD ELECTRONICS LABS.
- World Health Organization et al. (2010). Who laboratory manual for the examination and processing of human semen.
- Xie, J., Girshick, R., and Farhadi, A. (2016). Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning*, pages 478–487.
- Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5987–5995. IEEE.
- Xu, J., Luo, X., Wang, G., Gilmore, H., and Madabhushi, A. (2016a). A deep convolutional neural network for segmenting and classifying epithelial and stromal regions in histopathological images. *Neurocomputing*, 191:214–223.
- Xu, J., Xiang, L., Liu, Q., Gilmore, H., Wu, J., Tang, J., and Madabhushi, A. (2016b). Stacked sparse autoencoder (ssae) for nuclei detection on breast cancer histopathology images. *IEEE Transactions on Medical Imaging*, 35(1):119–130.
- Yang, J., Zhuang, Y., and Wu, F. (2012). Esvc-based extraction and segmentation of texture features. *Computers & Geosciences*, 49:238–247.
- Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

- Zeiler, M. D., Taylor, G. W., and Fergus, R. (2011). Adaptive deconvolutional networks for mid and high level feature learning. In *IEEE International Conference on Computer Vision*, pages 2018–2025. IEEE.
- Zhong, C., Han, J., Borowsky, A., Parvin, B., Wang, Y., and Chang, H. (2017). When machine vision meets histology: A comparative evaluation of model architecture for classification of histology sections. *Medical Image Analysis*, 35:530 – 543.
- Zohra, B. F. and Mohamed, B. (2009). Automated diagnosis of retinal images using the support vector machine (svm). *Faculte des Science, Department of Informatique, USTO, Algerie*.