



Melanoma Diagnostics Using Fully Convolutional Networks on Whole Slide Images

by

Adon Phillips

Thesis submitted in partial fulfillment of the requirements
for the Master of Computer Science degree

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa



This thesis is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).

ABSTRACT

Semantic segmentation as an approach to recognizing and localizing objects within an image is a major research area in computer vision. Now that convolutional neural networks are being increasingly used for such tasks, there have been many improvements in grand challenge results, and many new research opportunities in previously untappable areas.

Using fully convolutional networks, we have developed a semantic segmentation pipeline for the identification of melanocytic tumor regions, epidermis, and dermis layers in whole slide microscopy images of cutaneous melanoma or cutaneous metastatic melanoma. This pipeline includes processes for annotating and preparing a dataset from the output of a tissue slide scanner to the patch-based training and inference by an artificial neural network.

We have curated a large dataset of 50 whole slide images containing cutaneous melanoma or cutaneous metastatic melanoma that are fully annotated at $40\times$ objective resolution by an expert pathologist. We will publish the source images of this dataset online. We also present two new FCN architectures that fuse multiple deconvolutional strides, combining coarse and fine predictions to improve accuracy over similar networks without multi-stride information.

Our results show that the system performs better than our comparators. We include inference results on thousands of patches from four whole slide images, reassembling them into whole slide segmentation masks to demonstrate how our system generalizes on novel cases.

RÉSUMÉ

La segmentation sémantique comme approche de la reconnaissance et de la localisation des objets dans une image est un domaine de recherche majeur dans la vision artificielle. Maintenant que les réseaux de neurones convolutifs sont de plus en plus utilisés pour ces tâches, il y a eu de nombreuses améliorations dans les résultats de grand défis et de nombreuses opportunités de nouvelles recherche dans des zones autrefois inaccessibles.

En utilisant des couches entièrement connectées (FC), nous avons développé un pipeline de segmentation sémantique pour l'identification des régions des tumeurs mélanocytaires, de l'épiderme, et des couches du derme dans des images de microscopie complète de mélanome cutané et de mélanome cutané métastatique. Ce pipeline comprend des processus d'annotation et de préparation d'un ensemble de données recueillis à partir de la sortie de lames de tissus d'un scanner à la formation et à l'inférence basées sur les patchs par un réseau neuronal artificiel.

Nous avons organisé un grand ensemble de données de 50 images de lames microscopiques entières contenant de mélanome cutané ou d'un mélanome métastatique. Elles sont entièrement annotées à une résolution objective de $40\times$ par un expert pathologiste. Nous publierons les images de cet ensemble de données en ligne. Nous présentons également deux nouvelles architectures FC qui fusionnent plusieurs étapes, combinant des prévisions grossières et fines pour améliorer la précision sur des réseaux similaires sans l'information qui fusionnent plusieurs objets.

Nos résultats montrent que le système fonctionne mieux que celui de nos comparateurs. Nous incluons des résultats d'inférence sur des milliers de patchs à partir de quatre images de lames entières, en les réassemblant dans des masques de segmentation de lames complets afin de démontrer comment notre système généralise sur les nouveaux cas.

ACKNOWLEDGEMENTS

I must thank Prof. Lang for the time he dedicated to the many discussions we had during my time working on this thesis. His patience with me as I explored many ideas and his support for the final direction represented here is greatly appreciated.

I also express my sincere gratitude to Dr. Teo for the many hours she spent at the tedious job of annotating the source images as well as sharing her expertise freely and enthusiastically. Without her interest in this emerging technology and her eagerness to participate, this work would not have been possible.

Finally, the importance of my family's support goes without saying. There really are no words for me sufficiently express my gratitude to my wife Iris for her love, support, advice, and encouragement even as I spent increasing amounts of time working towards the completion of this manuscript. Thank you.

Table of Contents

List of Tables	ix
List of Figures	x
1 Introduction	1
1.1 Context	1
1.2 Background	2
1.3 Thesis Statement	3
1.3.1 Research Questions	3
1.3.2 Contributions	4
1.4 Thesis Organization	4
2 Related Work	6
2.1 Introduction	6
2.2 Convolutional Network Architectures	7
2.2.1 Basic networks	8
2.2.2 Fully Convolutional Networks (FCN)	9
2.2.3 GoogLeNet	11
2.2.4 ResNet	12
2.3 Segmentation Specific Review	14
2.3.1 Disease Detection in WSIs	14
2.3.2 Morphological Feature Segmentation in WSIs	14
2.4 Competitions and Winners	15
2.4.1 Camelyon	15
2.4.2 TUPAC	16
2.4.3 GlaS	16
2.4.4 BRATS	17

2.4.5	Pascal VOC	17
2.5	Dataset Details	18
2.6	Summary	19
3	Background	20
3.1	Introduction	20
3.2	Supervised Learning	20
3.2.1	Neural Networks	20
3.2.2	Regularization	24
3.3	Optimization	26
3.3.1	Gradient Descent	27
3.4	Backpropagation	28
3.5	Convolutional Neural Networks	34
3.5.1	Fully Convolutional Networks	43
3.6	Summary	43
4	Materials and Methods	44
4.1	Introduction	44
4.2	Dataset	44
4.2.1	Dataset Description	44
4.2.2	Source images	45
4.2.3	Case Selection Criteria	46
4.2.4	Gold Standard Annotation	46
4.2.5	Dataset Image Pre-processing	49
4.2.6	Data Augmentation	50
4.3	Network Architecture	56
4.4	Evaluation Metrics	62
4.4.1	Evaluation Procedure	64
4.5	Summary	64
5	Experiments and Results	65
5.1	Training Procedure	65
5.2	Network Selection Results	68

5.2.1	Network Selection Results Summary	76
5.2.2	Discussion of Experimental Results	76
5.3	Final Results	79
5.3.1	Discussion of Final Results	81
5.4	Summary	85
6	Conclusions	86
6.1	Introduction	86
6.2	Summary	86
6.3	Contributions	88
6.3.1	Limitations	89
6.4	Future Work	92
	APPENDICES	95
A	Pathologist’s diagnostic process	96
B	Background on Pathology Metrics	97
C	Tissue Variation	99
D	Anatomy of the Skin	100
E	Baseline for Scale	101
	References	103

List of Tables

5.1	Image details for our whole slide test cases.	69
5.2	Inference results for FCN-AlexNet	72
5.3	Inference results for FCN-16s	72
5.4	Inference results for Multi-Stride Fusion	72
5.5	Inference results for Multi-Stride + Multi-Scale Fusion	73
5.6	Performance metrics summary for each network	76
5.7	Settings used for primary training hyperparameters	78
5.8	Metrics for qualitative patch comparison	80
E.1	Evaluation results using the Multi-Scale-1 network	102

List of Figures

3.1	Diagram of a simple feed-forward neural network	22
3.2	Visualization of our simple linear network	30
3.3	Inputs and outputs of a single neuron	31
3.4	Example of the convolution operation	36
3.5	A visualization of some activation maps	37
3.6	An example of the max-pooling operation	39
3.7	Visualization of the ReLU function.	41
4.1	Example of feature and gold standard patches	50
4.2	Histogram of pixels by class before augmentation	53
4.3	Post augmentation histogram, stage 1	54
4.4	Post augmentation histogram, stage 2	55
4.5	Diagram of the processing pipeline	56
4.6	FCN-16s diagram	59
4.7	Multi-stride fusion diagram	60
4.8	Multi-stride, multi-scale fusion diagram	61
5.1	Overview of training and inference	66
5.2	Example of subtracting the dataset mean image	66
5.3	A visualization of the learning rate decay pattern	67
5.4	FCN-AlexNet training details	73
5.5	FCN-16s training details	74
5.6	Multi-stride fusion training details	74
5.7	Multi-stride, multi-scale fusion training details	75
5.8	Validation accuracy summary for each network	77
5.9	Qualitative comparison of results for selected patches	79
5.10	Whole slide image segmentation results	81

5.11	Example of tiling artifact in segmentation results	84
6.1	Variations in the dermis annotations	90
C.1	Tissue variation examples by type	99
D.1	Diagram of skin anatomy	100
E.1	Comparison of a baseline multi-scale network with FCN-16s	101

Introduction

1.1 Context

Cutaneous melanoma (CM, melanoma) is an aggressive form of skin cancer originating in melanocytes which are the cells responsible for the pigmentation of skin, hair and eyes. Melanoma develops as the result of DNA damage usually as a consequence of intense exposure to ultraviolet radiation. When a melanocyte suffers sufficient genetic damage it may begin to grow in an abnormal way and become a melanocytic tumour [1]. In the earliest stage of development the tumour will grow laterally along the outer epidermal layer of the skin (see Figure D.1 for a diagram of the skin anatomy). At this point the complete surgical excision of the lesion is possible. However, if left undiagnosed the tumour may thicken and infiltrate deeper into the skin. If the tumour growth invades lymphatic vessels, cancer cells may separate from the primary tumour and travel to regional lymph nodes as metastases in-transit. The tumour may also invade blood vessels causing cancer cells to be transported to distant locations resulting in metastatic cancer [2].

In the United States melanoma is the second most common cancer in men aged 20-39 and the fifth most common in men of any age. In women it is the third most common cancer through ages 20-39 and the sixth most common in women of all ages [3, 4]. In places such as New Zealand and Australia, melanoma incidence is reaching epidemic levels especially in young people [5]. Fortunately if diagnosed early, the five year survival rate is as high as 98%. If the tumour has spread to regional nodes, five year survival is reduced to 62% and then to only 18% in the case of metastatic disease [3]. While melanoma represents 3% of all skin cancers, it is responsible for 65% of skin cancer deaths [6]. Clearly the importance

of early detection and accurate diagnosis cannot be overstated.

The gold standard for the diagnosis of melanoma requires full-thickness excisional biopsy of suspected lesions followed by the histopathologic analysis of hematoxylin and eosin (H&E) stained tissue sections [7]. Historically, and to this day this analysis is conducted under a light microscope for the vast majority of cases. However the recent availability of slide scanning technology, which is able to digitize prepared tissue specimens on glass slides into high resolution files has brought the medical specialty of pathology into the midst of a transition towards a digital workflow. As a consequence, this move to “Digital Pathology” (DP) has also afforded the computational image analysis community access to whole slide images (WSI) allowing the application of modern image processing techniques to the processing of such specimens [8]. This includes automated, computer-aided, and other diagnostic tools that have the ability to augment current clinical workflows in the hopes of improving patient care and outcomes.

1.2 Background

A major component of the diagnostic process is a pattern recognition task where the pathologist uses visual information combined with deep domain knowledge to identify complex histologic and cytologic features. Machines too can be trained to become very good at recognizing patterns. There is much in the literature describing machine based techniques for object detection, segmentation, and other quantitative tasks on digital pathology images. Many of these techniques however, require expert input in the design of engineered or “handcrafted” feature descriptors that attempt to account for all variances that may be encountered in the specific problem domain. The process of engineering features can be time consuming and difficult to optimize for the target dataset, let alone to provide a generalizable solution. In the case of cancer diagnosis, the variance in slide preparation, nuances of anatomy relating to qualitative and quantitative criteria of diagnosis, not to mention variability introduced by different slide scanners and associated software processing creates a feature space that is beyond the capacity of most handcrafted descriptors/extractors to account for. This has constrained the practicality of machine learning for digital pathology.

In 1998, building on a vast mathematical progression of error minimization techniques in

complex, nonlinear, differentiable and high-dimensional parameter spaces [9], LeCun et al. introduced convolutional neural networks (CNNs). LeCun et al. combined an established learning method of gradient descent via back-propagation with their convolutional architecture that allowed the network to dynamically build feature maps directly from the annotated training data [10]. While very successful at the time, for many years the potential of this architecture lagged behind the availability of datasets with the requisite number of examples and more importantly, sufficient computational processing power. In 2012 a new era of “deep learning” was ushered in with the exceptional performance by Krizhevsky et al. [11] on the ImageNet ILSVRC-2013 [12] contest. Their work demonstrated that an efficient GPU implementation of the convolution and gradient descent operations made previously intractable or exceedingly expensive computations achievable in significantly less time on consumer level hardware.

The effective and robust feature learning capacity of CNNs is a natural fit for medical image processing problems. Despite the time-consuming and technically challenging task of dataset ground truth labeling, combined with limited availability of skilled annotators, state-of-the-art performance has been realized in a number of digital pathology tasks such as mitosis detection [13], breast cancer detection [14], glioma grading [15], epithelial and stromal segmentation [16] among others. However these successes are only the beginning of what is possible.

1.3 Thesis Statement

This thesis represents the details of our research exploring the application of machine learning methods to the challenge of cancer diagnosis from digital images. We develop the basis for a diagnostic pipeline for the annotation of gigapixel whole slide images to the detection of tumour cells and the segmentation of different tissue types relevant to diagnostic/prognostic factors of skin cancer. To our knowledge there has yet to be work published in the important area of melanoma diagnosis using deep learning approaches.

1.3.1 Research Questions

We address the following questions in this thesis:

1. How well can we train a machine to detect and segment melanocytic tumour cells in WSIs versus human experts?
2. How accurately can we train a machine to segment the epidermis and dermis layers, structures required for identifying two important diagnostic metrics known as Breslow thickness, and the Clark level? (see [Appendix B](#) for details on these metrics)
3. Using a trained model, can the segmentation inference be run on a WSI in a practical time-frame on consumer hardware?

1.3.2 Contributions

In seeking answers to the above research questions we have arrived at the following contributions:

1. We have assembled and annotated a new dataset containing 50 whole slide images of primary cutaneous melanoma or cutaneous metastatic melanoma.
2. We develop the basis of a slide understanding framework to characterize diagnostically important structures for the histomorphometry and staging of melanoma cases through semantic segmentation using fully convolutional neural networks. We also present an adaptation of this method for the detection of melanoma tumours in whole slide H&E pathology images.
3. We have developed a novel multi-stride fusion, fully convolutional architecture designed to mimic human approaches to processing the rich information present in a microscopic tissue specimen. We also adapt this architecture to include multi-scale information.

1.4 Thesis Organization

This thesis consists of 6 chapters. We provide a brief summary here:

In [Chapter 2. Related Work](#), we review the most important and applicable CNN architectures in the broad field medical imaging. We describe the state-of-the-art in deep learning methods for cancer detection in WSIs and comparable methods on a similar scale

to our approach in other malignancies. We review important challenges and their winning submissions.

In Chapter 3. *Background*, we provide an introduction to the mathematical basis of machine learning and further details regarding convolutional neural networks and their structural layers.

In Chapter 4. *Materials and Methods*, we describe in detail each step in the the development of the work presented herein. We describe the curation of the dataset including the acquisition and annotation of the data. We also provide motivations for the design of our proposed networks as well as their architectures and the basis of their comparison and evaluation.

In Chapter 5. *Experiments and Results*, we outline the results of our experiments. First we provide a comparison of our proposed architecture versus two industry standard designs. From these results we select the design for further training, based on which we provide the final semantic segmentation and detection results. We discuss the results and identify challenges and improvements.

In Chapter 6. *Conclusions*, we summarize this thesis and state our conclusions based on the results of the previous chapter. We describe how the work could be improved and taken further as future work.

Related Work

2.1 Introduction

In the past two years deep learning architectures and algorithms especially convolutional neural networks (CNNs) (see Chapter 3 Convolutional Neural Networks) have had a significant impact on the medical imaging community. As a result, a major new branch of medical imaging has emerged that is re-examining the landscape of many disparate techniques with attempts to find improvements for applicable areas using approaches based on deep neural networks. In some areas we are seeing new leaders appear in long held challenges and as of 2016 the state-of-the-art for many such challenges is defined by deep learning approaches. In this chapter, we review works that have applied deep learning techniques to the field of medical imaging. As technical contributions to the field are transferable, to a large degree to many areas beyond our specific focus, we have chosen to look across the field for inspiration and guidance. We view the field as being divided broadly into imaging modalities, application area and specific learning task. Modalities such as magnetic resonance imaging (MRI) appears to be the most common modality in works published in the last year, followed by digital microscopy and Computerized Tomography (CT). Other imaging modalities including ultrasound, X-Ray, and mammography among others were less common. The accumulated area of disease detection was the most common application, though the application areas specific to anatomical regions was also significant, with brain related works being the most common. Where possible we skew our review towards semantic segmentation approaches across application areas and modalities. This includes segmentation of structures such as whole organs, general anatomy, disease abnormalities, tumours, and other lesions.

To structure this review we have organized this chapter around common CNN architectures, as a description of the architecture is common to most works in this area. For reviewed works that use a “generic” architecture or the architecture was otherwise unimportant we include two sections to capture those works by task. Importantly, solutions to challenges across applications and modalities have a large degree of commonality also. Thus we review the larger challenges and their winners. Finally since the construction of our dataset was a major part of our work we include a section to capture the details of how other datasets were created. Throughout we use the term “segmentation” to mean “semantic segmentation”.

2.2 Convolutional Network Architectures

Advancements in classification, detection and segmentation tasks in medical imaging using CNNs are usually described in terms of their network architectures. The fundamental building block of all CNNs is simply the combination of a convolution function and a nonlinear function. An architecture describes various ways of stacking, combining, and connecting these structures with other structures such as pooling functions in a general or task specific way to minimize an error function when trained on a dataset (see [Convolutional Neural Networks](#) in Chapter 3). There are limitless ways of assembling a network, but recently some fundamental architectures have emerged that serve as the basis for incremental changes. These architectures have proven their utility in many state-of-the-art results and we describe their conceptual ideas and resultant exemplars, as they have greatly influenced our thinking, and in the case of the fully convolutional network (FCN), serve as the basis of the work in this thesis. Some architectures described are better suited for one of classification, detection or segmentation. While this work is primarily concerned with segmentation, architectures that succeed in detection or classification are also germane, as each are recognition tasks under which ideas cross-pollinate, and a network can generally be converted through minor changes to perform any of the three. Not all works we cover are explicitly based on any one of these specific architectures but all are based on the same evolution of ideas.

2.2.1 Basic networks

Here we provide a brief overview of two networks that could be considered the basis of all other networks covered in this chapter. In our work these networks were the basis for early training experiments which evolved into the final network used in this thesis.

The first CNN to gain prominence was “AlexNet” by Krizhevsky et al. [11]. Their work popularized CNNs in the computer vision community by winning the ImageNet ILSVRC 2012 challenge [12]. This network is simple by today’s standards but had a revolutionary impact on the field at the time and continues to be used and perform well. The network uses five convolutional layers, three fully connected layers, max-pooling and dropout layers (again please see Convolutional Neural Networks in Chapter 3 for more technical details). In the convolutional layer, they popularized the use of the rectified linear unit (ReLU) as the default nonlinearity function, as it decreased training time compared to the previous and much slower to compute *tanh* function. They were also the first to stack convolutional layers together without a pooling layer in between. They used extensive augmentation techniques combined with dropout layers to prevent overfitting. Krizhevsky implemented the network from scratch using CUDA (since there were no GPU based deep learning frameworks at the time). Due to the size of the ImageNet dataset a complicated pipeline was required to split the training over two GPUs to allow practical training times.

The first work to demonstrate the importance of network depth for improving performance was Simonyan and Zisserman’s VGGNet. This network placed second in the ImageNet 2014 (first in the localization task) contest by stacking sixteen convolutional layers using small 3×3 filters. This was a departure from the large filter of 11×11 used by Krizhevsky et al. in their first layer. They stacked smaller filters to reduce the number of parameters while they maintained a larger effective receptive field size when using back-to-back stacking (convolutional layers stacked together without a separating nonlinear layer). The authors continued the use of the ReLU function based on Krizhevsky et al. They also used data augmentation including scale jittering to prevent overfitting. The VGGNet became the basis of the fully convolutional network (FCN) for semantic segmentation tasks, perhaps due to its performance on localization tasks. Training the VGGNet for ImageNet took approximately three weeks on four NVIDIA Titan Black GPUs [17].

2.2.2 Fully Convolutional Networks (FCN)

One of the first works that used a deep learning architecture for segmentation of medical images was by Ciresan et al., who used a deep CNN to segment membranes in electron microscopy (EM) images. Their patch based, binary segmentation approach out-performed all other methods at the time. However their use of a computationally inefficient “sliding window” method, which computes convolutions for the same pixels multiple times due to the patch overlap, left room for improvement [18]. In 2015 Ronneberger et al. out-performed Ciresan et al. in segmenting neuronal membranes by dispensing with the “sliding window” approach in favor of their U-Net architecture. The U-Net is a modified FCN where the network structure uses a symmetric down-sampling then up-sampling path that aims to preserve context by retaining feature channels during up-sampling. The authors make extensive use of data augmentation techniques due to the small dataset available in the ISBI 2012 segmentation challenge (30 images of 512×512)[19].

The FCN architecture on which Ronneberger et al. based their U-Net was introduced by Long et al. [20]. Prior to their publication, classification networks were having great success in coarse inference, but the next step was clearly dense, pixel-wise prediction models. In a general classification network the fully connected layers discard spatial information as the order of input pixels will not affect the output order. To retain the spatial information, Long et al. replaced the fully connected layers in classification nets with convolutional layers. In doing so, the network computes a nonlinear filter down to the output map which contains the localized activations as predictions. At this point however, the predictions are in a down-sampled dimensional space of the input as a result of the convolution and pooling layers of the network. To up-sample the output map to the correct size, Long et al. use convolutional layers with a fractional input stride (combined with an activation function) to learn the up-sampling as a filter within the network (often called deconvolution) [21]. The ability to control the input stride for the deconvolution allows the network to learn up-sampling filters of selectable granularity. We exploit this ability to capture multiple levels of granularity and fuse the results as we will discuss in Materials and Methods in Section 4.3. Since Long et al.’s publication, FCNs have become the foundation for segmentation architectures in convolutional neural networks regardless of application domain[20].

FCN architectures brought localization to successful classification networks allowing dense inference, though methods of incorporating more context into the training process are emerging. One example of added context is the inclusion of multiple image scales in a multiple stream architecture or multi-scale convolutional networks (MSCN). Song et al. applied this concept in their cervical cytoplasm and nuclei MSCN to improve detection in difficult cases where pixel intensities are similar between distinct structures [22]. We also make use of multi-scale information in the design of our CNN architecture.

To suppress bones in chest X-ray (CXR) images to highlight soft tissue features, Yang et al. used a multi-scale network in the gradient domain (differences between neighboring pixels). They trained their network on different image scales in a cascade such that coarse information is propagated to the fine-scale in what are effectively three separate CNNs fused together. They train each of four levels separately before cascading the networks for inference. They evaluated their results quantitatively by comparing multi-scale and single scales networks as well as gradient domain vs intensity domain images by mean error. They also made qualitative evaluations comparing their output with a dual energy subtraction (DES) method where CXRs are taken with two exposures (one for bone, the other for soft tissue) and the bone highlights are subtracted to obtain a soft tissue image. The results were comparable, but do not require the additional hardware of the DES system [23].

Improvements in segmentation have been made by incorporating a conditional random field (CRF) with the FCN to provide more information to the network. In the case of Wang et al., in their segmentation of muscle tissue in the diagnosis of inflammatory bowel disease and so called “messy regions”, where important structures may have been disrupted during the process of preparing the slide, created a multi-scale FCN combined with a CRF which provides additional structural information. Like our work, this paper describes a multi-class segmentation problem (versus binary segmentation) on histology images. Their dataset included two hundred 1000×1000 images. The authors claim state-of-the-art results on their task. Interestingly their work showed that the CRF only offers improvements when the network is training on single scale images. Their multi-scale network performed better without CRF information, and better than the single scale network with CRF [24]. Our decision to not include a CRF in our network at this stage is partially based on this finding. Litjens et al. use a fully convolutional network to obtain whole slide level per-pixel likeli-

hood maps and segmentations for prostate cancer and breast cancer metastases in sentinel lymph nodes. In fact the network was trained using a regular CNN with fully connected layers, but was converted to a FCN for inference due to the significant performance improvements that can be realized with this approach. Overall their architecture was simple, containing only four convolutional layers in a single scale stack. The datasets included 254 prostate cases scanned at $40\times$ objective. The images were annotated with an in-house drawing tool to delineate cancerous regions. The authors selected a patch size of 128×128 pixels for training resulting in 920,000 patches. Dataset folding was stratified by patient [25]. Their work operates on WSIs and results are presented on the WSI level, consistent with ours. This work is similar to ours as we look holistically at the overall slide and not at selected regions.

2.2.3 GoogLeNet

While not a segmentation architecture per se, the GoogLeNet architecture has appeared in a number of state-of-the-art systems for tasks in medical imaging. As the trend towards deeper networks with more complex structures (versus simply stacking convolutional and pooling layers) gained momentum, Google introduced the inception block as part of the 22-layer GoogLeNet architecture. Their team was motivated by the idea that sparsely connected architectures will be important in deep learning given that network performance is proportional to the depth of the network, but where depth scales linearly, compute requirements scale quadratically. Each inception module in the stack composes average pooling with 1×1 , 3×3 , and 5×5 convolutional scales on a single image/patch. Importantly the 1×1 convolution + ReLU is applied before each larger convolution to reduce dimensionality (in the “depth” channel), thus minimizing what would be an explosion in the number of filters and giving superior performance with fewer parameters than a single convolution operation alone. To avoid the vanishing gradient problem, which becomes more of an issue in deeper networks, the authors use the combination of auxiliary softmax classifiers at multiple levels to compute the total loss during training. This has the added benefit of improving performance by incorporating early discriminative information in to the final classification. Overall their network used 9 inception models for a total of over 100 layers [26].

Inception models based on this architecture have emerged with state-of-the-art results on detection tasks in medical imaging. Liu et al., analyzing gigapixel WSIs from the Camelyon16 dataset (see subsection 2.4.1, and a 110 slide independent test set reduced the false negative rate in the detection of small tumours by 50% of the previous best. They used an inception architecture trained on multi-scale patches at two magnifications after experimenting with four. Due to the class imbalance in their dataset (2% median tumour positive patches per slide), the authors conducted extensive data augmentation operations including multiple orientation, colour, and jitter perturbations [27]. Other successful works demonstrating near human level performance on medical tasks include Gulshan et al., who used an inception based CNN to detect diabetic retinopathy using a massive dataset of over 125,000 photographs [28]. Also using a GoogleNet inception architecture Esteva et al. developed a CNN to classify images of external skin cancer lesions in 757 classes from 129,450 images. Their method is fast enough to be used on mobile devices in a clinical setting, while achieving accuracy performance equivalent to 21 dermatologists [29].

When experimenting with GoogleNet on our specific problem, we failed to achieve accuracy results comparable to a generic AlexNet model. This could be due to problems with our conversion of the network to fully convolutional.

2.2.4 ResNet

In keeping with previous findings that network depth is proportional to network performance, He et al., also inspired by VGGNet sought to push the limits of depth by developing a residual learning framework known as ResNet. At first the authors simply stacked convolutional layers, but had poor results and confirmed the existence of an accuracy degradation problem. As the network depth increases, at a certain point during training the accuracy becomes saturated before a dramatic degradation, resulting in higher error. To address this problem and allow for very deep networks (> 150 layers) the authors developed the residual block. They formulate the residual learning block which fits a residual mapping via shortcut connections that skip one or more layers and perform an identity mapping by adding their outputs to the outputs of the stack without adding trainable parameters to the network. This construction is modeled as:

$$H(x) := F(x) + x \quad (2.1)$$

If $H(x)$ is a desired mapping resulting from a few stacked layers and $H(x) - x$ is a residual function, then the authors hypothesized that it should be easier to optimize the residual mapping rather than the original mapping ($H(x) := F(x)$). Without this residual block the computation of the weighted convolutional step results in a new representation of x . However, with the identity x in the skip connection the block computes the required term $F(x)$ that must be added to x to obtain $H(x)$. This effectively finds some small perturbation to the input that satisfies the representation while maintaining some residual knowledge of the input (hence the name). The result of this, is a network that does not suffer from the degradation problem regardless of depth, and performs better than generic networks. This solution further emphasized the importance of skip connections, and made extensive use of batch normalization [30] to allow faster training and reduced overfitting. Their design won first place in both ImageNet and COCO challenges including COCO segmentation. Training for ImageNet required three weeks on eight GPUs [31].

For segmentation tasks on medical image datasets, combining ResNet architectures with FCN has produced state-of-the-art results and highlighted the potential for very deep networks for segmentation. Drozdal et al. built on their previous work of a fully convolutional ResNet (FC-ResNet) [32] to combine FCNs with ResNets, where the FCN stage learns to normalize the inputs, replacing the traditional preprocessing typically required by FC-ResNets. This FCN normalized data is then passed to the 100 plus layer FC-ResNet [33]. Their work achieves state-of-the-art performance on multiple datasets including PROMISE12 (prostate MRI segmentation) and their own in-house liver metastases CT dataset.

Schaumberg et al. used a 50 layer ResNet to predict the existence of a genetic mutation from visual features present on hematoxylin and eosin (H&E) stained WSIs. They pretrained their network on ImageNet, and extensively augment their images including six degree incremental rotations in addition to flipping, mirroring and cropping. As with our dataset, the authors sourced their images from The Cancer Genome Atlas (TCGA).

2.3 Segmentation Specific Review

2.3.1 Disease Detection in WSIs

There are a number of other works that seek to identify diseased tissue structures in whole slide images, but either use a generic CNN stacked architecture or do not highlight the architecture as an important part of their work. In preparing this thesis the following additional works in our specific domain were reviewed: Razaeilouyeh et al. used a CNN to detect breast cancer and perform prostate cancer Gleason grading. The authors applied a shearlet transform to the images and used the magnitude and phase as additional information for the network [34]. Kim et al. used a pretrained AlexNet for the classification of thyroid cancer according to a custom ontology used for searching for tumour images according to a semantic query within a mobile application [35]. Ertosun and Rubin used WSIs of glioma tumours from TCGA to train a network to grade such tumours. Their network is an ensemble of two CNNs where the first network differentiates between two types of glioma, then in the case of lower grade glioma (LGG), a second CNN performs a grading [15]. Chang et al. created a multi-scale convolutional sparse coding method to learn filters from biomedical images to improve transfer learning for medical tasks. The filters can be learned in an unsupervised way for training with larger numbers of images potentially allowing greater performance of networks that depend on transfer learning [36].

2.3.2 Morphological Feature Segmentation in WSIs

Most of the work in digital pathology involves the detection/segmentation of cellular structures such as nuclei, mitoses, individual cells and in our case epidermis/dermis layers of the skin. The motivation for identifying these structures relates to their role in various quantification methods as part of the diagnosis or staging of disease. Here we briefly outline additional works in this area that were reviewed as part of our work: Janowczyk et al. developed a resolution adaptive deep hierarchical learning scheme for nucleus segmentation [37]. Kashif et al. combined hand-crafted features computed using the scattering transform for texture based features with a CNN to detect nuclei in cells which performs better than CNN alone [38]. Turkki et al. created a method to detect tumour-infiltrating

lymphocytes by classifying superpixels. Their CNN method outperformed texture based features and was equivalent to agreement between two human pathologists [39]. Veta et al. used a CNN based regression model to measure the mean nuclear area without additional steps in 4000×4000 pixel images. They converted their network to a FCN for inference. Interestingly one might expect segmentation to be required but was avoided in this solution [40].

In a work with some direct application overlap with ours is that of Lu and Mandal who used traditional methods for the segmentation of the epidermis in WSIs. In their work the authors use a thresholding operation based on the probability density function of the red channel at an optimized level computed per image to obtain a binary image with the initial epidermis pixels in the foreground mask. Their process then eliminates false regions in the mask using application specific priors. This work is done on a down-sampled version of the target image to improve efficiency, since operating on the full-resolution image requires over three hours of processing time. Once the refined low resolution mask is complete, the mask is tiled according to the layout and matched with the full resolution image for further processing. They evaluate their method against manually annotated ground truth images [41].

2.4 Competitions and Winners

One of the major motivators in the area of digital pathology research has been the creation of grand challenges. These challenges have been important in encouraging the development of new techniques and establishing the state-of-the-art in their respective areas. Here we outline some of the major contests and their winners. These works were reviewed for insights on creating datasets and for standardized evaluation metrics. Details of evaluation methods will be covered in Evaluation Metrics in Chapter 4.

2.4.1 Camelyon

The Camelyon Grand Challenge, held by the International Symposium on Biomedical Imaging (ISBI) focuses on the detection of breast cancer metastases in WSIs of lymph

node sections. Participants have access to over 600GB (400 WSIs 270/130 split) of data allowing the submission of a broad array of architectures including very deep models. The winners of the latest contest in 2016 use the GoogLeNet architecture as the basis for their patch-based system. Their key insight was enriching the training set with difficult negative patches that were wrongly detected in previous experiments, as well as training a second network. Each network outputs a probability map that is thresholded and averaged together over connected components of the detected tumour to produce the final output [42].

2.4.2 TUPAC

The tumour proliferation assessment challenge (TUPAC) at MICCAI is based on mitosis (cells in the process of dividing) detection as an important prognostic marker in breast cancer diagnosis. There are three tasks in the challenge: Predict a tumour proliferation score from H&E stained tissue sections in WSIs by counting mitotic figures, predict a proliferation score based on molecular data, and detection of mitoses. The best performance for all tasks in the 2016 challenge came from Paeng et al. For their work, Paeng et al. based their network on the ResNet architecture to form an “L-View” architecture that mitigates the zero padding effect. The authors used a patch-based system for presenting input from WSIs to the network (due to the computational complexity required to handle complete WSIs). They converted the network to a FCN for inference [43].

2.4.3 GlaS

The GlaS Challenge Contest challenges submitters to automate the grading of colorectal adenocarcinoma through gland segmentation for the extraction of quantitative features. To enable this challenge Sirinukunwattana et al. published a dataset for gland segmentation in histology images of colon tissue. This dataset contains 165 WSIs from 16 patients scanned and processed to a $20\times$ objective magnification equivalent. The authors selected 52 visual fields from the WSIs containing a wide array of tissue structures and glandular architectures. The visual fields were annotated by an expert pathologist with boundaries for glandular objects and grading as either benign or malignant. The authors chose to

stratify images across training and test sets by “visual field” (an intermediate step of 52 images) as opposed to Janowczyk et al. who stratify by patient. Evaluation of submissions was based on detection accuracy, boundary similarity of glands, and volume-based accuracy of segmentations [44]. The best performing submission came from Xu et al., who used an FCN based architecture augmented with an instance segmentation capability developed by the authors, based on the concept of Holistically-Nested Edge Detection (HED) [45]. Their process involves the fusion of multiple “side-channel convolutions” to produce the edge-maps and regions that are concatenated to produce the instance segmentations [46].

2.4.4 BRATS

Menze et al. created a database of 65 images radiographic images, specifically MR scans to form the MICCAI Multimodal Brain Tumour Segmentation (BraTS) Challenge [47]. The challenge includes a task for the segmentation of glioma type of brain tumours. The training data (2017) consists of multi-type, multi-class image scans where each class representing a tumour subregion is annotated by an expert neuropathologist. The subregions are combined to form the ground-truth for the whole sample. The multi-class nature of the dataset is of interest to us as the data preparations are a critical aspect of such a project. In their solution to the 2016 challenge, Kamnitsasa et al. wanted to incorporate explicit global (location in the brain) and local information (detailed structures) as additional context for their model. To do this they created a dual pathway 3D CNN that simultaneously operates on normal resolution and downsampled input. Their method achieves state-of-the-art performance on the Brain Tumour Image Segmentation Benchmark 2015 (BRATS) and Ischemic Stroke Lesion Segmentation 2015 (ISLES). We note their use of a fully connected conditional random field (CRF) on the CNN’s output responsible for the final segmentation labels. [48].

2.4.5 Pascal VOC

To establish a basis for decisions in curating our multi-class dataset we looked to the more general field of semantic segmentation. The Pascal Visual Object Classes (VOC) Challenge [49] matched many of our criteria and was a major source of inspiration in how

our dataset was annotated. Everingham et al. used a colour mask for pixel level ground-truth annotation over 20 classes and 11,530 images containing 5034 segmentations. Borders between classes are the most challenging areas to provide pixel-accurate annotations in a time-efficient manner. To simplify the annotation of border regions, Everingham et al. employed a five pixel wide boundary stroke with a “void” label around each object. The void label represents objects that are neither object or background. The Pascal Challenge was last held in 2012, but their evaluation server remains available for public submissions. As of this writing the highest result was achieved by Lin et al., who used a combined CNN-RNN architecture for scene understanding. Their network generates hierarchical representations of the scene which allows a deeper understanding of the contained objects. They trained their network on extensive unsupervised or weakly-supervised sources [50].

2.5 Dataset Details

Research using machine learning, especially deep learning often takes a data-driven approach, sometimes requiring vast amounts of annotated ground-truth data for model training. Working with medical source images entails the additional challenges of working with gigapixel images in proprietary formats. Here we identify some works that required the curation of images for a new dataset, their details and the processing that was used. Despite the many works covered above we have found few works describing datasets of whole slide digital pathology images with multi-class semantic segmentation annotations. Most DP datasets for segmentation tasks use binary masks or contours to delineate the border of the target tissue type.

Janowczyk and Madabhushi discuss four segmentation works: The first task (nuclei segmentation), uses 143 images at $40\times$ magnification where each image is 2000×2000 pixels which are split into 32×32 patches. The second task (epithelium segmentation), uses 42 images at $20\times$ magnification where each image is 1000×1000 pixels which are split into 32×32 patches. The third task (tubule segmentation), uses 85 images at $40\times$ magnification where each image is 775×522 pixels which are split into 32×32 patches. Patches are down sampled to 10x equivalent magnification. The final task discussed (invasive ductal carcinoma segmentation), uses data from Cruz-Roa et al. [14] starting with 163 WSIs

scanned at $40\times$, then down-sampled 16:1 to generate 100×100 image patches, later resized to 32×32 by Janowczyk and Madabhushi. We note the WSI annotations in this case are very coarse high-level contours with low precision, drawn at 2x equivalent magnification [51, 14].

2.6 Summary

In this chapter, we have presented an overview of the the most influential works that contributed to this thesis. We positioned our work at the intersection of convolutional neural networks and multiple application areas within medical imaging. We develop our thinking on designing a convolutional neural network by reviewing the evolution of the field and how modern techniques are applied to medical imaging in various related application areas. We combined some existing ideas with our own concept to make improvements in our network architecture. Finally we found support for our decisions on dataset curation, and confirmed no prior work in our specific application area.

Background

3.1 Introduction

The purpose of this chapter is to provide the reader with a basic overview of the theoretical concepts of machine learning and convolutional neural networks that were important for development and implementation of this thesis. For a more comprehensive exploration of the fundamental concepts of neural networks we suggest the following excellent resources on the topic:

- “Pattern Recognition and Machine Learning” by Bishop [52]
- “Deep Learning” by Goodfellow et al. [53]
- “Elements of Statistical Learning” by Hastie et al. [54]
- “Large-scale machine learning with stochastic gradient descent” by Bottou [55]

The online course from Stanford University: CS231n Convolutional Neural Networks for Visual Recognition is also an excellent source of information [56].

3.2 Supervised Learning

3.2.1 Neural Networks

A neural network can be represented by a composition of n different functions, where a function $f^n(\cdot)$ represents layer n of a feed-forward network in the form:

$f(x) = f^n(f^{n-1}(\dots(f^2(f^1(x)))))$ where f^1 is the first layer and f^n is the n^{th} layer that serves to map an input X to an output Y as in $f : X \mapsto Y$. A layer can be composed of one or more neurons (often called perceptrons) that each compute a single output from a linear combination of the weights and real-valued inputs that are passed through a nonlinear activation function [57, 58]. A single neuron can be expressed mathematically as:

$$y(x, w) = \sum_j^N w_i x_i + b \quad (3.1)$$

where w , x , and b are vectors of the weights, inputs, and biases respectively. While a single neuron is limited to modeling linear functions, combining neurons into multiple layers with at least one hidden layer and a finite number of neurons allows any continuous function (in R^n) to be approximated when using a nonlinear activation function f as in:

$$y(x, w) = f \left(\sum_j^N w_i x_i + b \right) \quad (3.2)$$

This ability stems from the activation f that maps an otherwise linear output to a range of values on a monotonic, real-valued function. The term activation is a holdover from the biological inspiration of the perceptron as a model of the human nerve cell, but the analogy holds, as the range of possible values on a monotonic function indicate the magnitude of the output signal and thus the influence it has on the overall output, as in the action potential of a biological neuron. A network with multiple layers (often called a multi-layer perceptron) has a fully connected structure between layers, such that the neurons of layer $x + 1$ are connected to all the neurons of layer x . A simple neural network has three layers, consisting of a single input layer, one hidden layer, and a single output layer.

The weights used to compute each neuron's output are adjustable during the training procedure. These adjustments result in the network learning a set of weights that will satisfy the mapping $f : X \mapsto Y$ of an input image X to the desired output Y . Before a neural network can learn, we must provide it with a basis on which to build a representation

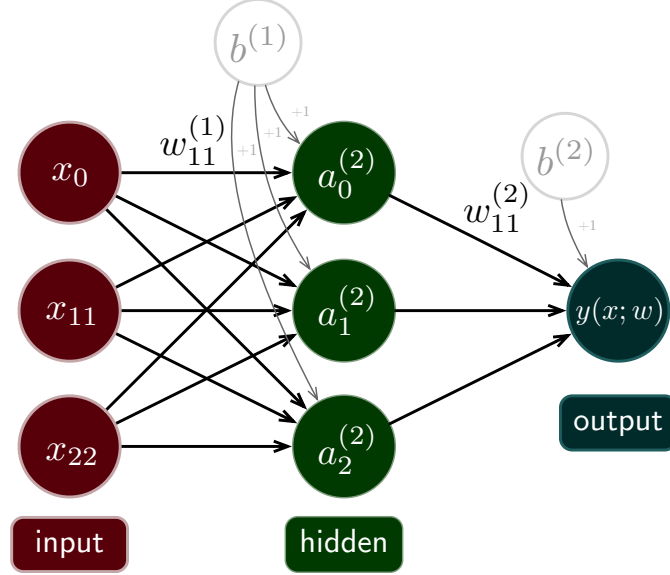


Figure 3.1 – A simple feed-forward neural network as an acyclic directed graph with weighted edges. The first hidden neuron computes the activation $a_0^{(2)} = f(w_{11}^{(1)} \cdot x_0 + b)$, $b = 1$. Finally the output returns the vector product of all weights and inputs plus the bias $y(x; w) = f((w)^T(v) + b)$.

of the space of images, in the case of a recognition type task, representative of the type that it will be asked to process. To do this, we provide the network with a set of n examples to be used in a training procedure to find the aforementioned mapping. In the case of supervised learning we must supply a ground truth value y with each training example x that matches our desired output $(x_i, y_i) \dots (x_n, y_n)$. We would like the function f to be learned automatically by the network and in the ideal case, that function will generalize to all values of X , not only those in the training set. This involves fitting a function via the optimization of an objective of the form:

$$f^* \approx \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n J(f(x_i), y_i) \quad (3.3)$$

where the lowest average loss across the training dataset is computed to minimize the expected value of a loss function J . In a neural network the **loss function** is the basis of learning. By mapping the weights of the network to a scalar-valued loss function $J(\hat{y}, y)$, we obtain a measure of how well the network performed by looking at the distance between a prediction $\hat{y}$ and the ground truth for a given input y . Using the loss function we can

iteratively determine the weights and bias values for the neurons in our network. To illustrate, consider the mean squared error (MSE) loss function J :

$$J(\theta) = \frac{1}{2n} \sum_x \|f(x) - a\|^2 \quad (3.4)$$

where θ is a vector representing all weights and biases in the network, $f(x)$ is the prediction output, and $\mathbf{a}$ is a vector of activations from the network for each input x in the training set of size n . From the equation we can see that the loss becomes small as $f(x)$ approaches a . This would indicate that the network has trained a model that may be a good fit for our data, and we say the network has converged.

After a model has been trained, its performance must be evaluated on test data that is similar to, but not included in the training set. The test data is randomly sampled from the dataset along with the validation and training data, such that the examples are independent and are sampled from the same probability distribution making them independent and identically distributed (i.e. they meet the i.i.d. assumption). This test set, like the training set contains data examples with their respective ground truths that can be used to determine how well the model generalizes on “new” data. A model that generalizes well has a low generalization error (sometimes called test error). During training the network performance is evaluated at regular intervals using the validation data. By observing these validation results hyperparameters can be tuned, and potential overfitting can be spotted allowing training to be stopped early (see Equation 3.2.2). There is no guarantee that a network will converge on a solution, or that the solution will generalize. Even with sufficient, high-quality training data, problems with the architecture of the network or the configuration of its hyperparameters (configurable variables that affect the critical aspects of the training procedure) can cause issues like overfitting or underfitting or simply prevent learning altogether.

Overfitting

The important part of a given dataset is the set of features that can be used to train an ideal model. If we are able identify this “signal” within the data we have the potential to train the best model. Unfortunately the dataset also contains “noise”, random variations

and other unimportant information which will not contribute to a good model. After training, if the model has learned both the signal and the noise in the data, the model has over-fit the data. Ultimately the training loss will be very low, but the test loss will be unacceptably high representing the models' inability to generalize.

Underfitting

It may be the case that the function that satisfies the mapping $f : X \mapsto Y$ is of very high complexity, but the design of the network is not capable of learning a function of commensurate complexity to sufficiently reduce the loss. Since it is trivial to increase the complexity of a model, underfitting is much less common than overfitting.

3.2.2 Regularization

A large challenge in learning a model is that the search space or hypothesis space for our loss function $f \in \mathcal{F}$ may be large with many possible solutions. For example, there exists a function where $f \in \mathcal{F}$ fits our mapping f perfectly for all training examples but suffers high loss for inputs not in the training set, thus does not generalize and has been over-fit. **Regularization** is an approach to constraining the search space of the loss function to reduce overfitting.

Under the premise that a less complex solution will generalize better than a more complex one with the same output loss, we can constrain the search space to favour less complex, lower order solutions. This approach, known as regularization, encompasses many strategies for reducing the test error while having a minimal effect on the training error and minimizing or eliminating overfitting. We can say that a regularized loss function usually generalizes better, but we caution that this is based on a simple heuristic, with no a priori reason to use regularization for every problem. We represent the regularization of our objective by adding a regularization term R to the loss function:

$$f^* \approx \arg \min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^n J(f(x_i), y_i) + \alpha R \quad (3.5)$$

where $\alpha \in [0, \infty]$ is a weighting parameter used to scale the effect of R which represents

the regularization strategy. Generally overfitting produces weight values whose magnitude is large, and R is formulated to penalize weights with large values in some way. Here we describe two common approaches to limiting the value of the weights [53]:

- **L^2 Regularization:** L^2 is a common form of regularization where the sum of the squares of the magnitude of each weight in the network is added to the loss function as $R = \sum_i w_i^2$ and scaled with α . This has the effect of increasing the error for large weights, thus leading to a diffuse set of weights consisting of many small weight vectors instead of fewer larger weight vectors.
- **L^1 Regularization:** L^1 is another common approach. Here the penalty is applied to the sum of the absolute value of each weight in the network. Again, this is added to the loss function in R but as $R = \sum_i |w_i|$ with the usual scaling from α . The result of L^1 functions as a feature selection mechanism by inducing sparsity into the weights profile where some weights have an optimal value of zero.

In addition to these methods, there are a number of regularization approaches that do not impact the loss function including:

Early Stopping

Overfitting is influenced by numerous factors, one of which is the overall training time. It is very difficult to estimate how long to train the network for optimal results or even to identify when overfitting begins during training. Early stopping is an approach used to identify overfitting as it happens, at which point training can be stopped (this can also be done retrospectively if network weights are captured and saved frequently during training). There are many variations of this approach [59, 53], though the general idea is to run an evaluation on the test data after each validation interval (i.e. every n iterations) to check for continued performance improvement or more simply take snapshots of the parameters frequently while monitoring the validation error, then returning to the snapshot with the lowest validation error. Ultimately, early stopping is widely used due to its simplicity and its ability to potentially reduce the computational cost during training. If used alone without L^1/L^2 regularization the computation cost of the penalty term and the potential tuning time is also saved.

Data Augmentation

One way to achieve some level of regularization is to add more raw data to the dataset.

This is one of the easiest and most effective forms of regularization. Training data (and associated annotation data for supervised training) is often limited, though it is possible to apply various transformations to the existing dataset to effectively generate new (x,y) exemplars for training. While it is not possible to apply augmentation to all types of source data, augmentation of image data can be particularly effective. Operations such as rotation, translation, and scaling are easy to implement and work well. Random cropping and occlusion masks can also result in improved generalization.

Dropout

Another effective approach to regularization, known as dropout, improves performance by averaging the outputs by geometric mean from what effectively becomes an ensemble of sub-networks within the overall base network. This method works by stochastically removing individual units, usually from hidden layers within a fully connected network to form a subnetwork where each forward propagation step runs through a different network for a given data example. By training on multiple “virtual” network architectures learned parameters are less dependent on other local parameter weights making overfitting less likely. Unlike combining results from discrete networks in an ensemble, parameters are shared between subnetworks since each is part of the same base network. As a result more possible subnetworks can be visited without exhausting available memory. Dropout can also be used with convolutional networks in convolutional layers (see section: 3.5 for details on convolutional networks). Despite the smaller number of parameters compared to fully connected layers, when added to lower layers dropout improves generalization by adding noise to the output activation maps, making them robust to image variations [60].

3.3 Optimization

In general terms, the process of learning involves the minimization of a loss function including the minimization of a regularization term. So learning becomes an optimization problem that requires some mathematical machinery within our neural network to perform. Here we will describe the gradient descent optimization algorithm.

3.3.1 Gradient Descent

The goal of the learning process is to find the weights for each node in the network that minimize the generalization error. This is an optimization problem that can be solved by moving the values of the weights in the direction of the (negative) slope of the derivative of the loss function. For a simple convex problem, with a few variables we might directly find the solution numerically via finite differences [61]. In this case, for each dimension of our parameter vector θ we can find the direction of the gradient by approximating the derivative via:

$$\frac{dE}{d\theta} \approx \frac{J(\theta + h) - J(\theta)}{h} \quad (3.6)$$

where h is the learning rate (or step-size). The parameter values can then be shifted towards the minimum. Given that a typical neural network's parameters number from tens of thousands to billions in large convolutional networks and form a loss function with a highly non-uniform error surface, this method becomes intractable at this scale. However, this method can be used to evaluate the gradient for non-differentiable loss functions. Also, it is common to use this approach, known as gradient checking, to evaluate the accuracy of implementation code that uses the calculus based method we will now discuss.

In lieu of a tractable numerical method, there exists an approach known as gradient descent that can be used to find the gradient in an iterative manner under the constraint of requiring a differentiable loss function. The gradient $\nabla_{\theta} J$ as a vector of partial derivatives of J (loss) with respect to the parameters (θ) of the network, provides a direction in which to move the parameter vector towards lower loss by stepping each parameter in the direction of the negative gradient. The step-size in this context is an implementation hyperparameter (learning rate) that has great influence on how the gradient descent algorithm proceeds. A sufficiently small step-size will guarantee a decrease in the loss value, but if the step-size is too small, the network may take too long to converge. Likewise with a step-size that is too large, the minimization will not converge even with infinite training time.

The gradient descent algorithm visits each training sample before taking a single step. In the case where the training dataset is large, the time cost may become too high to

wait for convergence. A modified version of the algorithm known as **stochastic gradient descent** (SGD) avoids this bottleneck by stochastically estimating the gradient based on the average gradient of a small subset (minibatch) of the full training dataset on a single update cycle. This has the effect of significantly increasing the number of parameter updates while reducing the time to convergence at the expense of not finding the true gradient due to the noise introduced when sampling examples for the minibatch [53]. It is important to note that SGD is more sensitive to dataset bias, requiring that minibatches contain a random set of samples. To counter this, datasets should always be shuffled prior to ingestion into the database or network to remove any bias that might arise from similarities between images that were otherwise stored with some kind of ordering.

Some improvements can be made to SGD to improve convergence under tricky conditions on the loss surface (especially long deep valleys). One method known as **momentum** or classical momentum accelerates gradient descent by introducing an accumulating velocity vector term that operates in the direction(s) of persistent descent. Momentum is defined as:

$$\mathbf{v}_{t+1} = \mu \mathbf{v}_t - \eta \nabla_{\theta} J(\theta_t) \quad (3.7)$$

$$\theta_{t+1} := \theta_t + \mathbf{v}_{t+1} \quad (3.8)$$

where $J(\theta_t)$ is the loss function, η is the learning rate and μ is the momentum coefficient that determines the rate of decay of the previous gradient's effect on v . A higher value indicates more previous gradients have influence on the current direction. This can be thought of as increasing the step-size proportional to the alignment and size of the sequence of previous gradients directions [62].

3.4 Backpropagation

Now that we have an algorithm to minimize the gradient of the loss function we need a method to efficiently evaluate the gradient of the loss with respect to each parameter in the network. Backpropagation is an efficient algorithm for computing derivatives in a

multi-layer neural network (see Figure 3.3 for a high level visualization).

The algorithm has three basic steps: a propagation step consisting of a forward pass to compute the loss J and backward pass to compute the gradient of the loss ∇J . Finally the parameters are updated via gradient descent using ∇J from the backward pass according to the learning rate.

Forward Pass

First the forward pass computes the output values for each neuron and the resulting network prediction. If a given neuron j has i inputs each with a weighted edge from $i \rightarrow j$, the output of the neuron j , denoted as y_j can be evaluated as the linear combination of weights and inputs forming its activation a when passed through a nonlinear, differentiable function f :

$$y_j = \sum_i w_{ij}x_i + b \quad (3.9)$$

$$a_j = f(y_j) \quad (3.10)$$

where f is a scalar valued function in $\mathbb{R}$ and b is the bias term. Generally we would like to compute the forward pass as a matrix product of the inputs and the weights. Using a simple linear layer such that $Y = f(X, W)$ as an example without biases or activations:

$$X = \begin{bmatrix} x_{11} \\ x_{21} \end{bmatrix}, W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix} \quad (3.11)$$

where X is a matrix containing input minibatch of 1 vector of size 2 (1×3) and W is a matrix of weights containing two row vectors of size of size 3 (2×3). We represent this layer visually in Figure 3.2.

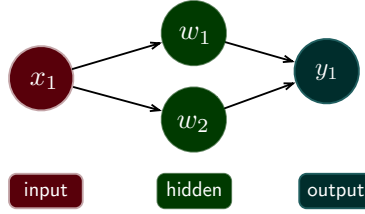


Figure 3.2 – Visualization of our simple linear network

On the forward pass we compute the output of each neuron $Y^T = X^T W$ as a simple linear combination of X and W :

$$Y = \begin{bmatrix} x_{11}w_{11} + x_{12}w_{21} \\ x_{11}w_{12} + x_{12}w_{22} \\ x_{11}w_{13} + x_{12}w_{23} \end{bmatrix} \quad (3.12)$$

The output Y would contain the real-valued confidence scores to be used as input to the loss function where the scalar loss J is calculated. Finally to prepare for the second step, the partial derivative of the loss J with respect to Y is computed for each element of Y :

$$\frac{\partial J}{\partial Y} = \begin{bmatrix} \frac{\partial J}{\partial y_{11}} & \frac{\partial J}{\partial y_{12}} & \frac{\partial J}{\partial y_{13}} \end{bmatrix} \quad (3.13)$$

In some implementations it is convenient to compute local gradients on the forward pass and keep them in memory for use during the backward pass. Since the local gradient is the partial derivative of the neuron's own output with respect to its own input(s) we have the required values on the forward pass.

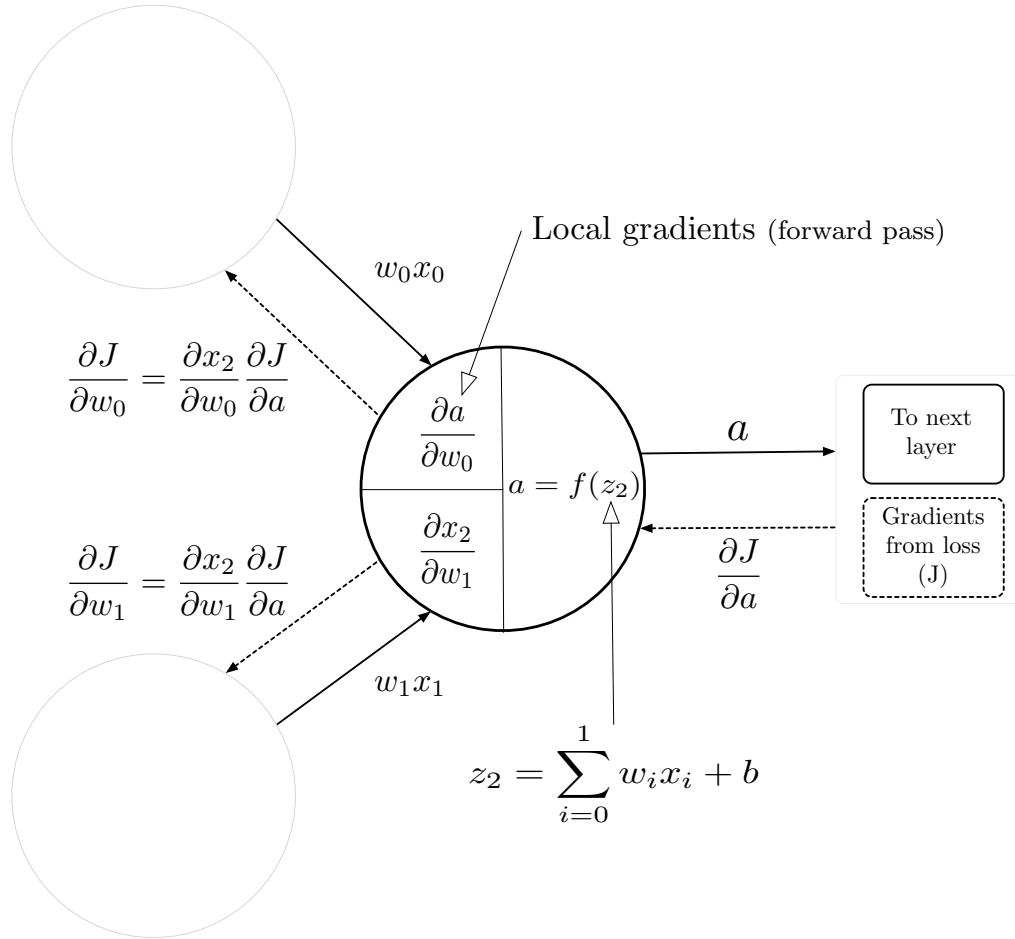


Figure 3.3 – A single neuron in a hidden layer showing inputs, outputs, and weights on the forward pass and the basic gradient calculations during backpropagation. The activation function is omitted in our example [63, 56]

Backward Pass

In the backward pass the final output loss is propagated back through the network to determine the influence that the output from a given neuron has on the final output loss, so that each neuron's parameters can be adjusted in the direction that will minimize the output. This is significant because a given neuron is part of a complex computational graph, where on inspection it is not clear how changing a parameter will affect the overall loss, making manual updates impossible. Backpropagation starts from the output (top) of the network and proceeds backwards to the input (bottom), recursively building up this influence signal so that the next neuron receives the gradient that encodes the influence of all previous neurons that the current neuron contributed to in the forward pass. Using our simple example, on the backward pass we compute $\frac{\partial J}{\partial X}$ using $\frac{\partial J}{\partial W}$. Here we use the chain

rule from calculus to find the loss with respect to the weights and the loss with respect to the inputs:

$$\frac{\partial J}{\partial W} = \frac{\partial J}{\partial Y} \frac{\partial Y}{\partial W} \quad (3.14)$$

$$\frac{\partial J}{\partial X} = \frac{\partial J}{\partial Y} \frac{\partial Y}{\partial X} \quad (3.15)$$

Now we will compute the partial derivative of J with respect to X (Note that we can use the same process for $\frac{\partial J}{\partial W}$, but for our example we focus on $\frac{\partial J}{\partial X}$). We have:

$$\frac{\partial J}{\partial X} = \begin{bmatrix} \frac{\partial J}{\partial x_{11}} & \frac{\partial J}{\partial x_{12}} \end{bmatrix} \quad (3.16)$$

then we compute the partial derivative of each element:

$$\frac{\partial J}{\partial x_{11}} = \frac{\partial J}{\partial Y} \frac{\partial Y}{\partial x_{11}} \quad (3.17)$$

$$\frac{\partial J}{\partial x_{12}} = \frac{\partial J}{\partial Y} \frac{\partial Y}{\partial x_{12}} \quad (3.18)$$

Now finding the derivative of Equation 3.12 with respect to X , we find the following:

$$\frac{\partial Y}{\partial x_{11}} = \begin{bmatrix} \frac{\partial y_{11}}{\partial x_{11}} \\ \frac{\partial y_{12}}{\partial x_{11}} \\ \frac{\partial y_{13}}{\partial x_{11}} \end{bmatrix} = \begin{bmatrix} w_{11} \\ w_{12} \\ w_{31} \end{bmatrix} \quad (3.19)$$

$$\frac{\partial Y}{\partial x_{12}} = \begin{bmatrix} \frac{\partial y_{11}}{\partial x_{12}} \\ \frac{\partial y_{12}}{\partial x_{12}} \\ \frac{\partial y_{13}}{\partial x_{12}} \end{bmatrix} = \begin{bmatrix} w_{21} \\ w_{22} \\ w_{23} \end{bmatrix} \quad (3.20)$$

Having $\frac{\partial J}{\partial Y}$ and $\frac{\partial Y}{\partial X}$, both with shape 1×3 :

$$\frac{\partial J}{\partial x_{11}} = \frac{\partial J}{\partial Y} \cdot \frac{\partial Y}{\partial x_{11}} = \frac{\partial J}{\partial y_{11}} w_{11} + \frac{\partial J}{\partial y_{12}} w_{12} + \frac{\partial J}{\partial y_{13}} w_{13} \quad (3.21)$$

$$\frac{\partial J}{\partial x_{12}} = \frac{\partial J}{\partial Y} \cdot \frac{\partial Y}{\partial x_{12}} = \frac{\partial J}{\partial y_{11}} w_{21} + \frac{\partial J}{\partial y_{12}} w_{22} + \frac{\partial J}{\partial y_{13}} w_{23} \quad (3.22)$$

Finally substituting back into Equation 3.16:

$$\frac{\partial J}{\partial X} = \begin{bmatrix} \frac{\partial J}{\partial x_{11}} & \frac{\partial J}{\partial x_{12}} \end{bmatrix} = \begin{bmatrix} \frac{\partial J}{\partial y_{11}} w_{11} + \frac{\partial J}{\partial y_{12}} w_{12} + \frac{\partial J}{\partial y_{13}} w_{13} \\ \frac{\partial J}{\partial y_{11}} w_{21} + \frac{\partial J}{\partial y_{12}} w_{22} + \frac{\partial J}{\partial y_{13}} w_{23} \end{bmatrix} \quad (3.23)$$

$$= \begin{bmatrix} \frac{\partial J}{\partial y_{11}} & \frac{\partial J}{\partial y_{12}} & \frac{\partial J}{\partial y_{13}} \\ \frac{\partial J}{\partial y_{21}} & \frac{\partial J}{\partial y_{22}} & \frac{\partial J}{\partial y_{23}} \end{bmatrix} \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} \quad (3.24)$$

$$= \frac{\partial J}{\partial Y} W^T \quad (3.25)$$

We see that by solving the partial derivative of the output Y with respect to each weight we can simplify $\frac{\partial J}{\partial X}$ to $\frac{\partial J}{\partial Y}$ multiplied by W^T . As a result, there is no need to form the Jacobian when we compute $\frac{\partial J}{\partial W}$ making the implementation much more efficient. The same can be done for $\frac{\partial J}{\partial X}$ with the approach we have shown here such that $\frac{\partial J}{\partial W} = X^T \frac{\partial J}{\partial Y}$ [56, 64].

Weight Update

Once the gradients have been computed for each neuron, each weight can be shifted in the direction of the gradient flow using the gradient descent algorithm. The new weight value will be set according to the learning rate.

Please note that this example was used to illustrate the general concept of the backprop-

agation algorithm. While the structure holds for more realistic examples, consider the following:

- We would likely have a higher order tensor as input, changing our indices from scalars to vectors.
- When computing the outputs with respect to the inputs we must consider the derivative of the activation function.
- We would need to propagate across multiple layers by including the activations of the next (i^{th}) layer e.g. $\frac{\partial J}{\partial y_j} = \frac{\partial Y}{\partial y_i} \frac{\partial y_i}{\partial x_i} \frac{\partial x_i}{\partial y_j}$.
- Backpropagation in convolutional layers requires a more advanced treatment.

Here we summarize the overall training algorithm:

Algorithm 1 Stochastic Gradient Descent via Backpropagation

Input: Training Set $(x_i, y_i) \dots (x_n, y_n)$

Initialize: Parameter weights.

- 1: **repeat** Assemble minibatch of n items from training set.
 - 2: **for each:** example x in minibatch **do** ▷ Forward Pass
 - 3: Compute $z_j = \sum_i w_{ij}a_i + b$ and $a_j = f(z_j)$
 - 4: Compute local gradients δ
 - 5: Estimate $\nabla_J \approx \frac{1}{n} \sum_{i=1}^n Jy(f_w(x_i), y_i) + \alpha R(f_w)$ ▷ Backward Pass
 - 6: Backpropagate the error computing: $\frac{\partial J}{\partial w_{ij}}$
 - 7: Use gradient descent to update the weights according to $W^{i+1} = W^i - \eta \nabla_{w^i} J$
 - 8: **until** all training examples processed.
-

3.5 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a special class of neural network that implement a series of architectural constructs to dramatically reduce the number of learnable parameters compared to fully connected nets. This makes it possible to train networks using higher order tensors with large dimensionality such as using images as inputs that would be intractable for fully connected networks whose practical use is limited to images on the order of 200×200 pixels on consumer hardware [56]. Rather than use a 2D mesh of neurons the CNN transforms inputs into volumes of neurons with sparse interconnectivity.

The sparsity arises from the local connectivity with specific regions of the input that gives rise to spatial awareness within the network that is important for image processing.

A CNNs like any feed-forward neural network, finds the mapping $f : X \mapsto Y$. In a CNN, the intermediate functions of the mapping are represented by a set of layer types that perform transformations on 3D volumes towards the final output. There are many potential layer types, though generally a layer will be derived from one of the following types:

- Convolutional Layer: Convolves local weights as filters over the input
- Nonlinear Transformation: Applies a fixed activation function
- Pooling: Performs a downsampling operation on the input
- Fully Connected: Produces a classification score

When arranged in a sequence these layers form a convolutional neural network that may be trained using SGD via backpropagation. The network itself and the individual layers have hyperparameters that allow certain configurable properties of the layer to be adjusted prior to training. We now provide an overview of these layer types:

Convolutional Layer

The convolutional layer implements the convolution operation, that in the context of neural networks is an expression of the visual overlap between two tensors, usually images. One image being the input, and the other being a set of learnable weights known as a kernel or a filter. A convolutional layer may contain many filters each having a small spatial extent and a depth matching the input volume. We can describe the convolution operation mathematically for a 2D input:

$$C(i,j) = (I * F)(i,j) = \sum_m \sum_n I(i+m, j+n) \cdot F(m,n) \quad (3.26)$$

where the $*$ operator represents the convolution operation. Note that this differs slightly from the conventional definition in that the filter is not flipped relative to the input (as in

$\sum_m \sum_n I(i-m, j-n) \cdot F(m, n)$). This has the effect of removing the commutative property from the operation, but more importantly flipping the kernel changes the placement of the parameters which is undesirable in a learning context. The equivalent operation but for a 3D input volume:

$$C(i, j, k) = (I * F)(i, j, k) = \sum_m \sum_n \sum_o I(i + m, j + n, k + o) \cdot F(m, n, o) \quad (3.27)$$

During the convolution operation the filter slides over the image and at each step the dot product between each element of the filter matrix and the local area of the image that maps to the filter's current position is computed (see Figure 3.4 for a visualization of this operation). The result is a single channel activation map (also called a feature map) indicating the regions of the input that responded to the filter (see examples in Figure 3.5). Each filter associated with a given convolutional layer is convolved against the input in the same manner and the results are sequentially stacked in the depth channel of the output volume such that if there were 64 filters in the layer, the output would also have a depth of 64. The number of filters, their size, and the filter's step-size or stride across the input are hyperparameters that have a substantial impact on the overall training output.

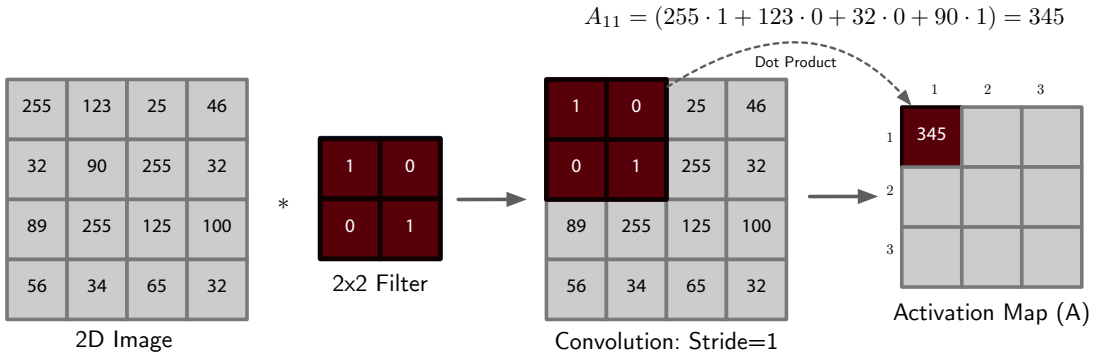


Figure 3.4 – Example of convolution operation when the element-wise dot product between the filter and the overlapping region of the image is computed. Using a 2×2 filter on a 4×4 image with a stride=1 results in a downsampled activation map of shape 3×3 .

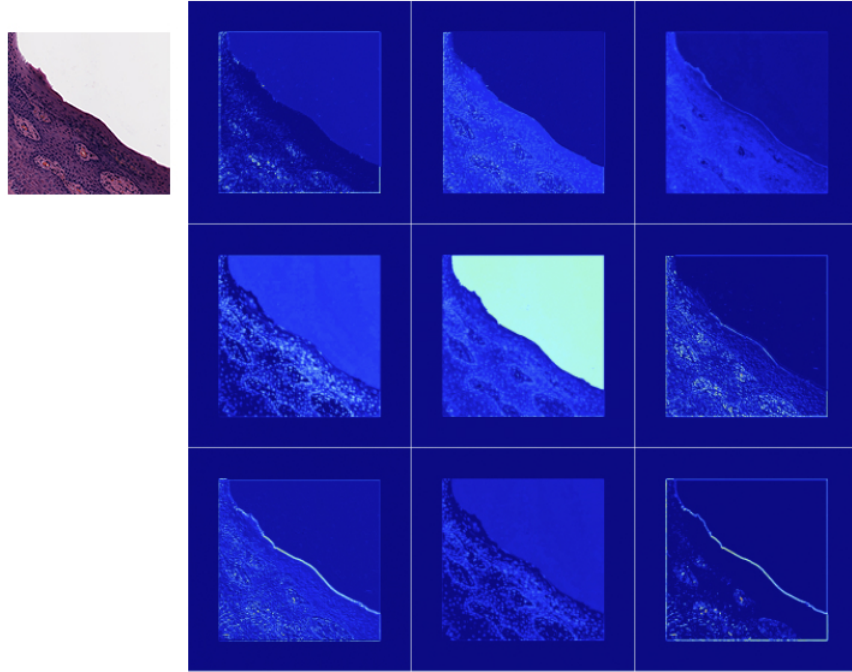


Figure 3.5 – A visualization of activation maps taken following the first convolution block of the network shown in Figure 4.7 in Chapter 4. The activation shows areas in the images that responded to the various filters used in the convolutional layer. Lighter colours represent areas that were “excited” by a given filter. Looking at the example in the center of the grid, the “background” area is excited after having been convolved with a filter of a similar colour to the same area in the input.

Compared to feed-forward networks, convolution is highly advantageous from a compute perspective. Since the filters are significantly smaller than the input, and convolutions are associated with localized blocks of pixels, the result is sparse connectivity compared to the dense connectivity in feed-forward nets. Take an input image of shape $200 \times 200 \times 3$ for example; in a fully connected network with 20,000 neurons in the first hidden layer this would mean $120,000 \text{ input neurons} \times 20,000 \text{ hidden neurons} = 2,400,000,000$ parameters in the first layer. Compare this to learning filters in a convolutional layer, where with the same input and 64, 3×3 filters, using a stride of 4, and without zero-padding (adding zeros to the border of the input volume to control the spatial size of the output volume) according to:

$$\text{Convolutional layer output size} = (W - F - 2P)/S + 1 \quad (3.28)$$

$$= ((200 - 3 - 2(0))/4) + 1 \quad (3.29)$$

$$= 50 \quad (3.30)$$

where in one of equal dimensions W denotes the input size, F denotes the filter size, P represents the zero-padding, and S represents the stride, we end up with $50 \times 50 \times 64 = 160,000$ neurons in the first layer multiplied by $3 \times 3 \times 3$ (+1 bias for each filter) parameters in each filter results in $160,000 \times 28 = 4,480,000$ parameters - which would be less CPU and memory intensive to compute, but is still too many for efficient computing, even on modern GPUs, considering we generally need multiple convolutional layers. This can be substantially improved by sharing the parameters across each depth channel. Each of our 64 filters represents a unique feature detector (e.g. a vertical edge in an early convolutional layer) gradually learned during training. We can imagine that these features could be found anywhere within the input's spatial extent. So instead of learning 64 new features for each of the 50×50 regions of the layer, we share the 64 features across all regions. Otherwise using our example, we would be asking the network to learn a new vertical edge detector for each of the 50×50 neurons along with the 63 other feature types. Using this type of parameter sharing dramatically reduces the number of parameters resulting in a total of $64 \times (3 \times 3 \times 3 + 1) = 1792$ parameters.

Pooling Layer

Pooling is a form of dimensionality reduction used to reduce computational requirements in a CNN. A pooling layer is usually placed following a single, or a stack of convolutional layers. The pooling layer is used to downsample the activation maps in the volume output by the preceding convolutional layer. Downsampling the spatial dimensions of each activation map results in a new representation with increased invariance to small translations in the input and with significantly less parameters to be learned than without pooling. For example by reducing the width and height of an activation map by a factor of 2 the resultant contains 4 times less data. The reduction in computational complexity and reduced

memory footprint comes at the expense of some spatial information about local regions of the input being discarded, but in most cases is considered a worthwhile trade-off.

There are multiple pooling algorithms, but perhaps the one most often encountered is the max-pooling version (see Figure 3.6), where the highest activation in the region of the pooling filter is taken to represent the $n \times n$ area of the filter in the output. Other versions include stochastic pooling, and average pooling among others.

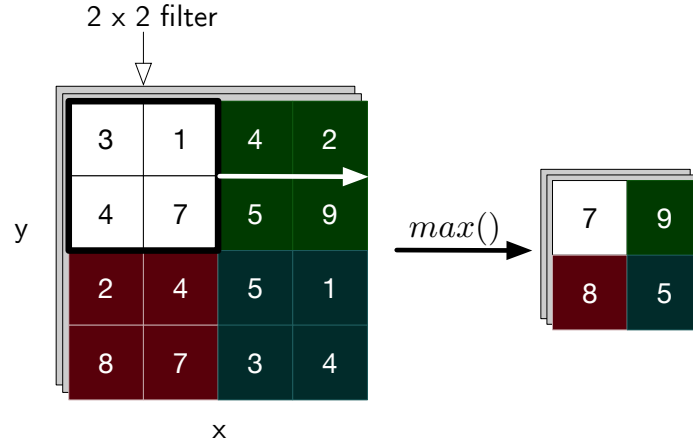


Figure 3.6 – Example of the max-pooling operation using a filter size of 2×2 and a stride of 2. Notice the input of 4×4 is downsampled by a factor of 2 resulting in a 2×2 spatial extent in the output volume. The $\max()$ operation returns the largest value from the set of inputs.

An alternate form of dimensionality reduction is the 1×1 convolution [65] which operates in the feature space depth volume. Where the standard pooling layers reduce the width and height of the input, a 1×1 convolution reduces the number of features (sometimes called feature pooling). Since the 1×1 convolution is usually followed by some nonlinear activation function the overall transformation is learned by the network, as seen in the Google Inception architecture. This adds spatial information to the transformation [26]. These convolutions can be combined with max-pooling layers to reduce dimensionality in three dimensions.

Fully Connected Layer

A layer is fully connected (FC) if the neurons of layer $x + 1$ are connected to all the neurons of layer x as in multilayer perceptron (MLP) feed-forward networks. When used

in convolutional neural networks for classification, the FC layer takes an activation volume as input and outputs a real-valued confidence score that indicates how well an image fits each class type the network was trained on. The scores are stored in a vector of size equal to the number of classes in the training dataset. Generally a softmax function is used to force the scores to sum to one in order to provide a probability distribution across the mutually exclusive classes.

Nonlinear Activations

A convolutional layer, or a stack of convolutional layers is generally preceded by an activation layer. The activation function $f(\cdot)$ enables the network to fit more complex and expressive models than would be possible with simple affine transformations that neurons compute internally. This is consistent with our intuition when observing semantic objects in images and the real world that are made up of nonlinear relationships (e.g. the relationship between an object's size and weight).

There are many functions that serve this purpose, however in the context of deep learning with CNNs, the rectified linear unit (ReLU) has been shown to be computationally efficient and able to avoid problems like the vanishing gradient that affected historically popular functions like the sigmoid function. ReLU is able to avoid these issues due to the form of its derivative, its lack of an expensive-to-compute exponential term, and near-linear graph (i.e. two linear components) that make it easier to optimize.

In the case of the sigmoid function defined as $\frac{1}{1+e^{-x}}$, while its input is squashed into a range between 0 and 1, its derivative ($\sigma' = \sigma(x) * (1 - \sigma(x))$) used to find the gradient, squashes the input to a maximum of $\frac{1}{4}$ with an asymptote at 0. As layers are stacked the overall maximum gets exponentially smaller. During backpropagation when we multiply derivatives of sigmoid based activations the output becomes small very fast leading to a **vanishing gradient**, causing the learning process to stop.

The ReLU function defined as $ReLU(x) = \max(0, x)$ (see figure Figure 3.7) also squashes the input, but into an integer value of either 0 or 1, thus the vanishing gradient problem is avoided since the derivative is always either 0 or 1:

$$\frac{d}{dx} \text{ReLU}(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ \emptyset & \text{if } x = 0 \end{cases} \quad (3.31)$$

when the input is less than 0 the derivative is 0, and when the input is greater than zero the derivative is 1. A small caveat is that the derivative of ReLU is undefined for input activations of zero. However, in practice the activation is rarely equal to zero, since SGD is unlikely to arrive at a zero value [53].

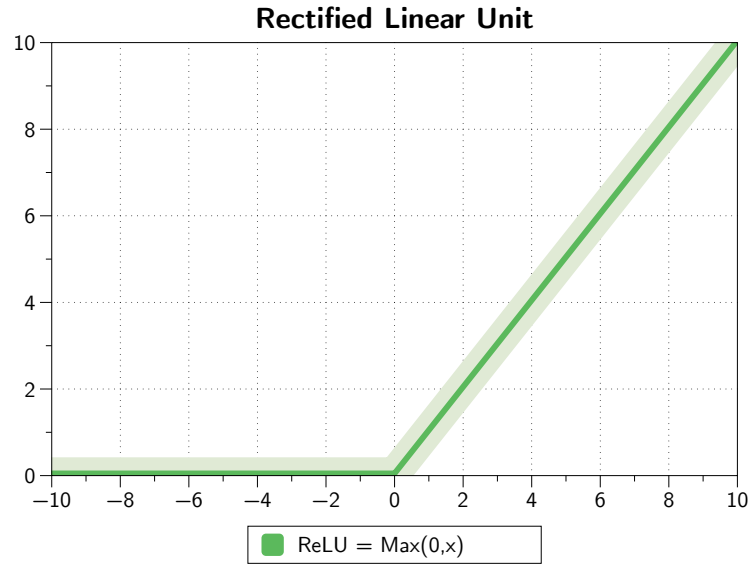


Figure 3.7 – Visualization of the ReLU function.

Example Architecture

The following list summarizes the “AlexNet” architecture [11], which through its success in the ImageNet challenge in 2012 has become a reference architecture for understanding CNNs. This architecture uses the building blocks outlined above to produce a network that is capable of training useful models:

- **Layer 0** Input image
 - Input Size: $227 \times 227 \times 3$

- **Layer 1** Convolution: 96 filters, size 11×11 , stride 4, padding 0
 - Output Volume: $55 \times 55 \times 96$
 Computed based on: $((227 - 11)/4 + 1 = 55)$ (see Equation 3.28)
- **Layer 2** Max-Pooling: 3×3 filter, stride 2
 - Output Volume: $27 \times 27 \times 96$ $((55 - 3)/2 + 1 = 27)$
- **Layer 3** Convolution: 256 filters, size 5×5 , stride 1, padding 2
 - Output Volume: $27 \times 27 \times 256$ padding $= ((5 - 1)/2 = 2)$
- **Layer 4** Max-pooling: 3×3 filter, stride 2
 - Output Volume: $13 \times 13 \times 256$ $(27 - 3)/2 + 1 = 13)$
- **Layer 5** Convolution: 384 filters, size 3×3 , stride 1, padding 1
 - Output Volume: $13 \times 13 \times 384$ padding $= ((3 - 1)/2 = 1)$
- **Layer 6** Convolution: 384 filters, size 3×3 , stride 1, padding 1
 - Output Volume: $13 \times 13 \times 384$ padding $= ((3 - 1)/2 = 1)$
- **Layer 7** Convolution: 256 filters, size 3×3 , stride 1, padding 1
 - Output Volume: $13 \times 13 \times 256$ padding $= ((3 - 1)/2 = 1)$
- **Layer 8** Max-pooling: 3×3 filter, stride 2
 - Output Volume: $6 \times 6 \times 256$ $(13 - 3)/2 + 1 = 6)$
- **Layer 9** Fully Connected: 4096 neurons
- **Layer 10** Fully Connected: 4096 neurons
- **Layer 11** Fully Connected: 1000 neurons
 This hold the final classification containing 1000 scores, one for each of the ImageNet categories.

3.5.1 Fully Convolutional Networks

In the case of dense prediction tasks like semantic object recognition or segmentation where each pixel in an input is assigned a class label, fully convolutional networks (FCNs) outperform traditional methods that use hand-crafted features and CNN models with learned features combined with some dense prediction step such as fully connected layers [66]. An FCN has the same structure as a CNN except the fully connected layers have been replaced by convolutional layers. FCNs can take an input of any size (only constrained by GPU memory), and where a standard CNN computes a nonlinear function, an FCN computes a nonlinear filter [20]. The primary benefit of this approach is the preservation of spatial information, which is important for localization in segmentation tasks that would be lost in the fully-connected layers.

3.6 Summary

In this chapter we have outlined the theoretical building blocks of neural networks and provided an overview of convolutional networks. We have described the mathematical basis of the major computational elements supporting the work of this thesis.

Materials and Methods

4.1 Introduction

This chapter will describe the resources and processes used to answer the major research questions addressed by this thesis. First we will describe the process by which we acquired slide data and assembled the dataset. We provide details regarding the annotation process, as it represents a critical aspect of how the system performs. We also include the motivation behind, and the process by which we arrived at our network architecture and the details of its design. Finally, we outline the metrics used for the evaluation and the process by which it was conducted.

4.2 Dataset

4.2.1 Dataset Description

Our dataset includes 50 whole slide images (WSIs) containing hematoxylin and eosin (H&E) stained, full-thickness excisional biopsies from 49 individuals. Source images were scanned at 40 $\times$ objective magnification using Aperio ScanScope slide scanners from nine different medical institutions. Annotations are provided at the equivalent resolution to the 40 $\times$ magnification source images. The overall dataset is divided into training, validation, and test sets at a ratio of 70:15:15 percent. The data set organized by patient to allow stratification by patient such that pixels from a given patient only appear in one of the training, validation, or test datasets.

Additionally, the annotation source files are included in Adobe Photoshop (Photoshop) PSD format to enable the addition of future tissue type annotations and other improvements. Using these files, patches of arbitrary size can be generated as requirements demand.

The source images for the dataset were obtained from The Cancer Genome Atlas (TCGA). The TCGA is a project of the National Institutes of Health (NIH) in the United States that has made over 2.5 petabytes of cancer related data available to the public including thousands of digital pathology images including approximately one thousand cutaneous melanoma slides from the human skin cutaneous melanoma (HSCM) project, all available in the public domain without restrictions or limitations.

In accordance with the publication guidelines of the TCGA we hereby make the following declaration: The results shown here are in whole or part based upon data generated by the TCGA Research Network: <http://cancergenome.nih.gov/>.

4.2.2 Source images

The unannotated source images received from TCGA were saved in Aperio SVS format which is a standard format used by slide scanner manufacturers. The files contain multiple sub-files including the following:

- A large thumbnail of the overall slide
- A table of metadata identifying file and scanner information
- A full resolution TIFF file of the tissue region of the slide
- A series of pyramid images at multiple resolutions
- A slide label image used to track the slide in a clinical environment

For this thesis only the full resolution TIFF image was required. Using Imagemagick (<http://www.imagemagick.org/>), the sub-files in the SVS can be accessed with array notation on the filename (e.g. *slide_id_code.svs[2]*) for extraction to a standalone TIFF file. The TIFF files were extracted using a python script to create a standalone “uncompressed” TIFF and a second compressed version for import into Photoshop. This compression was required to reduce the file size to a level that enabled the images to be manipulated on

screen. Note that the “uncompressed” version was reserved for training (we use uncompressed in quotes as there is likely some compression used but not specified).

Within Photoshop, individual layers were used for each class. The required layers were generated, colour coded, and named automatically upon opening the file using Photoshop’s internal scripting engine. The compressed source file was placed at the bottom of the layer stack and the annotation layer made transparent.

4.2.3 Case Selection Criteria

The one thousand HSCM slides were manually filtered at very low resolution to identify H&E stained specimens containing epidermis and tumour cells. From this review 150 slides were identified for possible inclusion.

From the 150 candidate slides, inclusion was based on four main criteria:

1. To ensure more uniform quality of staining (to the degree possible) and to facilitate interpretation, material which appeared to be formalin-fixed paraffin-embedded tissue was selected over material with frozen section artifact (the dataset does not include any tissues from frozen sections).
2. The histology needed to be interpretable; tumour cells needed to be readily distinguishable from histiocytes, lymphocytes, and other stromal cells.
3. Large tumour sections were favoured over small biopsy specimens to maximize the number of labeled pixels.
4. Slides with higher proportion of uninvolved epidermis were also favoured, again in an attempt to maximize the amount of epidermis available for annotation.

4.2.4 Gold Standard Annotation

Colour mask, gold standard annotations were applied over the source images by an expert pathologist. The annotations were manually defined using three colors in the RGB colour space representing three tissue classes. Red (**#FF0000**) denoting tumour regions, yellow (**#FFFF00**) denoting the dermis layer, and green (**#00FF00**) denoting the epidermis.

The overall annotation also includes two additional colours to indicate a background class and a “not diagnostically important” (NDI) class. These are denoted by white (#FFFFFF) and black (#000000) respectively. The NDI class includes all tissue areas (including artifacts on the slide e.g. pen markings) not marked by the pathologist. The annotations are only an approximation of the ground-truth since we can not perfectly label every pixel and some pixels do not clearly conform to any of the available classes used in this work.

When testing annotation methods, the epidermal layer in the images introduced some challenges due to its undulating curvature and small proportionality compared to the other tissue types. Using contour based annotation methods was more time consuming and error prone due to the large number of points required to draw the rete ridges of the epidermis (finger-like structures of the epidermis). Also, when downsampling the images to reduce annotation time, even a reduction from 40 $\times$ objective to 20 $\times$ objective made the epidermis boundaries too poorly differentiated to provide an accurate annotation. Additionally simply loading, navigating, and saving the WSIs was often lagged and slow, even in software designed for viewing and annotating large files. As a result of these challenges, as mentioned, Adobe Photoshop was used for the annotations. Using Photoshop allowed the class’s annotations to be drawn on individual layers with the advantage of using Photoshop’s advanced manipulation tools. A set of manipulations was defined as a series of steps, switching between tools, to quickly label complex tissue areas. For example to label the epidermis, the “wand tool” could be used with a particular setting to select the majority of related pixels, then by expanding the selection, and using the “fill tool” those pixels could be labelled “automatically”. About five such workflows were established, along with a number of ad-hoc techniques that greatly decreased annotation time. As an added benefit, the performance of Photoshop was much better than other task specific tools, the mode of navigation across the image was familiar, and annotations could be made at any effective magnification arbitrarily using the “zoom tool”. The use of these advanced tools adds a “semi-automated” component to the workflow that offsets the additional annotation time required when working with larger resolution images.

The annotations were made using the full-size images with a 40 $\times$ objective resolution. The source images over which the annotations were made were compressed with the JPEG codec with $Q = 25 \Rightarrow 23 : 1$ ratio. Given that many of the slide images are on the order of 50,000² pixels the compression was required to manage file size. The 23 : 1 compression

ratio was sufficient to maintain the required level of detail for accurate annotations, but large enough to sufficiently reduce file size. Photoshop source files (PSD) with five layers plus the compressed source images as the base layer, with each layer having an area on the order of five billion pixels would result in a source file size between one and four gigabytes.

Annotation strategy

The task of labelling tumour regions was not as straightforward as initially anticipated. Like most tumours, melanomas are predominantly composed of proliferating, genetically damaged cells. However, these proliferative cells also elaborate signalling molecules which stimulate the remodeling and growth of vascular and other supporting tissues which supply it with a steady supply of nutrients and growth factors to maintain its constituent cells. Without these nutrients and growth factors, areas of the tumour may undergo cell death (necrosis) or other forms of degenerative change. While melanomas may try to evade immunologic detection, sometimes they evoke a host immune response, which appears as a thick rim or a heavy population of tumour infiltrating lymphocytes. Areas of degeneration may also stimulate a response from neutrophils and macrophages, among other inflammatory cells [67, 2]. All of these morphologic features are indicative of tumour presence were included in the definition of tumour regions.

As most of the selected WSIs contained tumours that were mass forming, the external edge of the tumour was outlined. Large areas which were composed of inflammatory cells were not marked as tumour; however, if melanoma cells resided within the inflammatory population and could be positively identified, these individual cells were marked. The fine blood vessels and stroma of the tumour were also included with the tumour, as were tumour infiltrating lymphocytes within the borders of the tumour. Necrotic areas, apoptotic cells, and any areas of cystic degeneration were also marked as tumour. At ulcer sites, only tumour cells were marked, and areas of fibrinopurulent exudate were not marked as tumour. Where they could be positively distinguished from mimickers, melanoma cells within the epidermis (in-situ melanoma) were also marked but were not actively sought out.

On some slides, tumour fragments within lymphatic or small thin-walled vessels were highlighted on a separate layer as lymphovascular invasion for possible inclusion in future work.

The epidermis was labeled from the basement membrane to the top of the granular layer (the outer edge of the epidermis). Markedly reactive epidermis and tangentially sectioned

epidermis was included. The stratum corneum (the outermost layer of the skin) was not labelled. As the infundibular portion of the hair follicle derives from the epidermis, this was included in the epidermis markings. In spite of its morphologic similarity to the epidermis, the remainder of the hair follicle was included within the dermis layer as it represents a dermal structure. This decision requires further analysis to determine the impact on the prediction of the included tissue types (see Figure D.1 for anatomical details).

The definition of the dermal layer provided another conundrum. While the dermis was normal in many cases, in others, scar or alteration by tumour was found. All of these were eventually included in the definition of the dermal layer. The lower boundary of the dermal layer was also difficult to define, as it undulates and merges imperceptibly with the fibrous septae of the underlying subcutaneous adipose tissue and an arbitrary cutoff had to occasionally be employed.

4.2.5 Dataset Image Pre-processing

Completed annotations were exported from Photoshop, combining class layers into a five colour, paletted image in PNG format with lossless compression. The exported files retained the case id portion of the original filename and were stored in a directory of the same name. The original full resolution source image was placed in the same directory along with the Photoshop source file retaining the discrete layers for future refinements and the addition of new tissue type annotations.

Patches

While our network is capable of processing input images of arbitrary size, the WSIs were much too large for wholesale processing on the GPU so it was necessary to crop the source images for patch based training (see Figure 4.1 for example patches). Source images and annotation masks were processed into 1024×1024 , 512×512 , and 256×256 patches for training and experimentation. The paletted annotation images have a five colour palette index, but splitting those files into patches results in various palette (also known as a colour map) arrangements across the patches due to nuances of the PNG encoder. In order to use the patches for training each patch must conform to the same colour palette index order regardless of the number of colours in a particular patch. To correct this a script was required to manually specify the same colour palette in the same order for each image.

Of course since the order of the palette changes, the pixel indices must be remapped to the new palette order. This requires every pixel of every patch to be visited which is a time-consuming process. Patches created from source TIF images were converted to lossless PNG format during the patch creation process. Since the pixel dimensions of the WSIs were not divisible by the patch dimensions, this resulted in misshapen patches along the top and right-most edges. These patches usually contained only background pixels and were discarded to ensure only uniform square patches were included. Patch files were divided into “annotations” and “features” directories with the respective annotation and feature patches stored with the same file name. Patches retain the case id filename with an incremented patch number suffix e.g. *TCGA-BF-A5EO-01Z-00-DX1_0.png*.

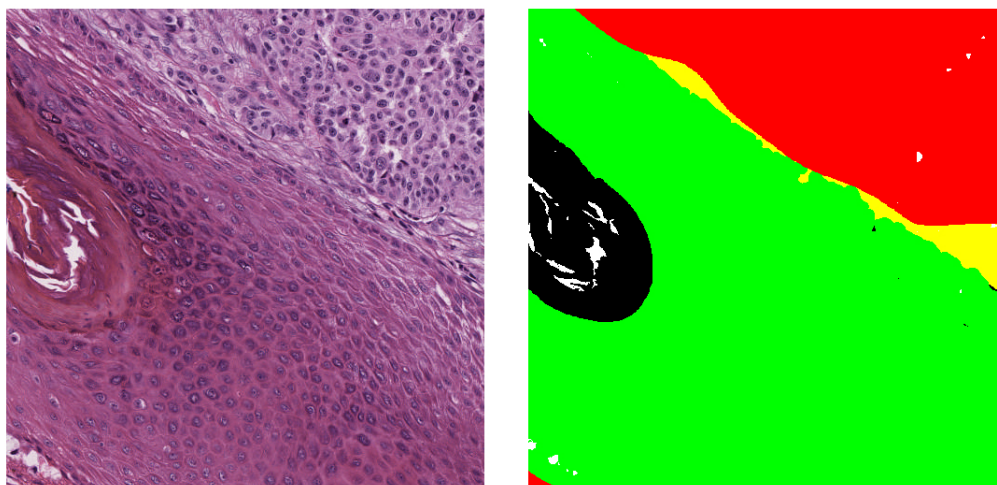


Figure 4.1 – An example of a source image patch with all tissue types present (left) and the corresponding gold standard annotation mask patch.

4.2.6 Data Augmentation

The thirty six patient cases included in the training set translates to over 62,000 image patches at 512×512 pixels or approximately 16.3 billion pixels. Based on other similar works using semantic segmentation [51, 14, 50, 47], this represents a sufficiently large collection of data for experimentation. However, as is the case with many medical imaging datasets, due to the natural distribution of the tissue types in our dataset, there exists a class imbalance between certain tissue types and between tissue types and the background.

We investigated multiple approaches to class balancing for the dataset. In cases where the imbalance is severe it is common to adapt the loss function to include class weights. In their MRI based tumour segmentation network, Christ et al. [68] added a weighting factor in the cross entropy loss function of their FCN as follows:

$$L = -\frac{1}{n} \sum_{i=0}^N \omega_i^{class} \left[\hat{P}_i \log P_i + (1 - \hat{P}_i) + \log P_i (1 - P_i) \right] \quad (4.1)$$

where P_i is the probability of belonging to the foreground and $\hat{P}_i$ represents the ground truth. In this case the imbalance is much more severe than ours at near 100:1. This approach generally has the unfortunate side-effect of introducing an additional hyperparameter that could be time-consuming to tune.

Another common approach is to perform an undersampling of the majority class to bring the classes into alignment. This method can be problematic as there may be information lost that could be important for learning, though in cases where a sufficient number of samples are available it can be a workable technique [69]. In cases where data is scarce, Yen and Lee [70] proposed a cluster-based approach to selecting samples. The main idea is to divide the dataset into k clusters and choose the clusters that will best represent the majority class according to criteria specific to the dataset at hand. Rahman and Davis [71] used a variation of Yen and Lee’s approach to balance a challenging cardiovascular dataset.

As an alternative to undersampling, especially in cases of limited data, oversampling the minority class may be appropriate. In this case the dataset distribution can be modified by duplicating training examples towards a dataset-appropriate metric [72]. The result is that the minority class will have a similar number of examples as compared with the other classes in the dataset. The derivation of the “duplicate” examples varies according to the problem. Where the data representation allows, it is very common to use synthetic examples generated according to some scheme such as k-nearest neighbors (kNN). In this case a randomly selected kNN is chosen and the feature vector difference corresponding to a target minority example is multiplied by a random number between 0 and 1 as in:

$$x' = x_i + (\hat{x}_i - x_i) \times \delta \quad (4.2)$$

where x_i is the target example, $\hat{x}_i$ is a randomly selected kNN of x_i and $\delta \in R$ is a random number between 0 and 1. This approach is known generally as the Synthetic Minority Oversampling Technique (SMOTE) [73]. Krawczyk et al. used a similar method to balance their breast cancer malignancy dataset where the number of normal samples greatly outnumbered the number of malignancy containing cases [69].

When it is not possible to generate synthetic examples that are known to accurately represent the spatial ordering and distribution of data, especially pixel data, it is common to sample examples from each class equally. This comes with the risk of biasing the result towards the rare class [48]. In their mitosis detection network Cireşan et al. mitigated this risk by first training their CNN without oversampling, then in a second stage oversampling the pixels that were incorrectly detected in the first stage [74].

Our dataset suffers from two major problems. Firstly, since WSIs consist of a tissue specimen placed in the center of the slide, and due to the background being visible through the tissue specimen, background pixels are the most common by far. Background pixels outnumber the second most common class (tumour) 4:1. Most of these pixels are in patches that only contain background pixels. Secondly tumour labelled pixels outnumber epidermis labelled pixels 28:1 (see Figure 4.2). Initial experiments with the unmodified data showed a test network would converge with high accuracy by predicting the tumour class for all tissue types (severe overfitting). Furthermore, early experiments also showed there were insufficient pixels labelled in the epidermis class to train the network to recognize such regions accurately in the face of such a large imbalance.

To address these problems we have formulated a hybrid class balancing solution to effectively undersample and oversample the problematic classes as required:

Undersampling

As stated, most of the background pixels are found in patches containing only background, from areas around the periphery of the WSIs. We undersample by removing patches where background pixels are most common and make up more than 98% of the target patch.

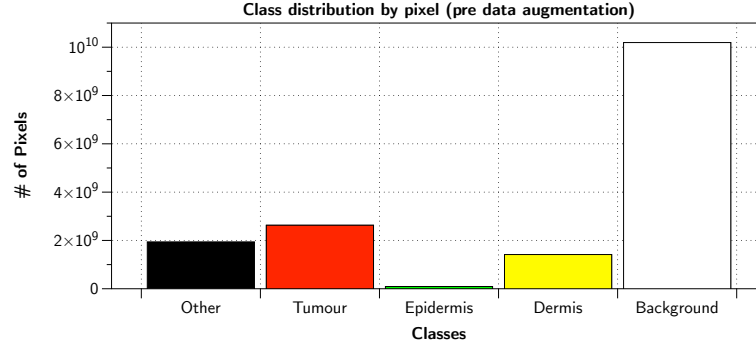


Figure 4.2 – A graphical representation of the class distribution by the number of pixels per class before data augmentation.

This percentage retains patches that might contain locally important features. We also remove patches where background pixels are most common and the NDI class is the only other class present.

Oversampling

For the epidermis class which is the minority class, we oversample using basic data augmentation transformations to increase the number of epidermis containing patches with respect to patches of other classes. We parse the pixels of each patch containing the annotations and store the list of patches that meet the following criteria:

- All pixels are epidermis
- Most pixels are background, second most pixels are epidermis
- Most pixels are background, second most pixels are NDI, third most pixels are epidermis
- Most pixels are NDI, second most pixels are epidermis
- Most pixels are NDI, second most pixels are NDI, third most pixels are epidermis

These annotation patches and the corresponding feature patches are then augmented with the following transformations:

- Flip left to right
- Flip top to bottom
- Rotate 90°
- Rotate 270°

Other transformations such as affine shifts, magnification changes, addition of noise, and lighting/intensity changes were avoided due to the potential that such changes were not consistent with the range of characteristics we would expect in real-world images containing epidermis.

Finally as an additional experiment, a source image containing a relatively large number of epidermis pixels was selected then cropped and annotated for only the epidermis class. This image was then augmented using the scheme described above and added to the training set.

The result of the proposed balancing scheme is a generally balanced dataset with a modest but tractable imbalance remaining between the tumour and epidermis classes. This imbalance was reduced from 28:1 to 7.75:1 (see Figure 4.3 and Figure 4.4). It might seem to make sense to undersample the tumour class to further reduce the imbalance, though after experimentation we found that the epidermis was receiving accurate segmentations compared to pre-balancing tests. Furthermore the anatomical variance of tumour cells and regions across patients and even individual slides is significantly greater than that of the epidermal cells (see Figure C.1 for an illustration of in-class variation). Therefore we thought it advantageous to retain as many tumour examples as possible.

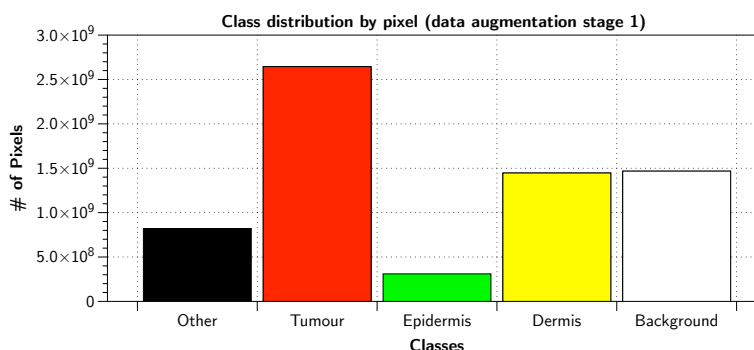


Figure 4.3 – A graphical representation of the class distribution by the number of pixels per class following the execution of the data augmentation strategy.

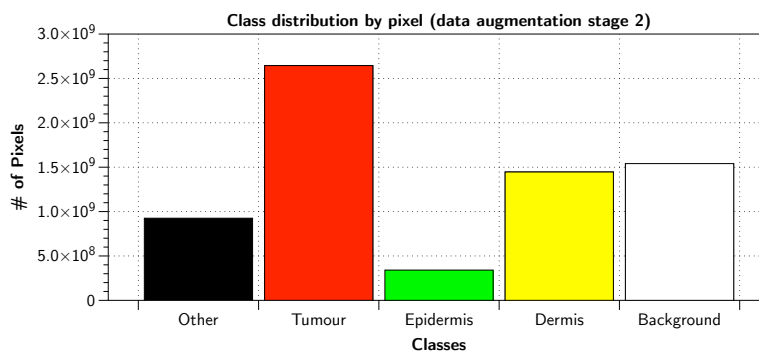


Figure 4.4 – Class distribution by the number of pixels per class following the described data augmentation strategy showing the addition of new epidermis only patches (minimal effect).

Dataset Processing Pipeline

In this chapter we have described the components of our WSI processing pipeline, beginning with source images as input, through to the resulting training, validation, and test datasets composed of patches of three distinct resolutions as output ready for the training process. We provide a visualization of this workflow in Figure 4.5

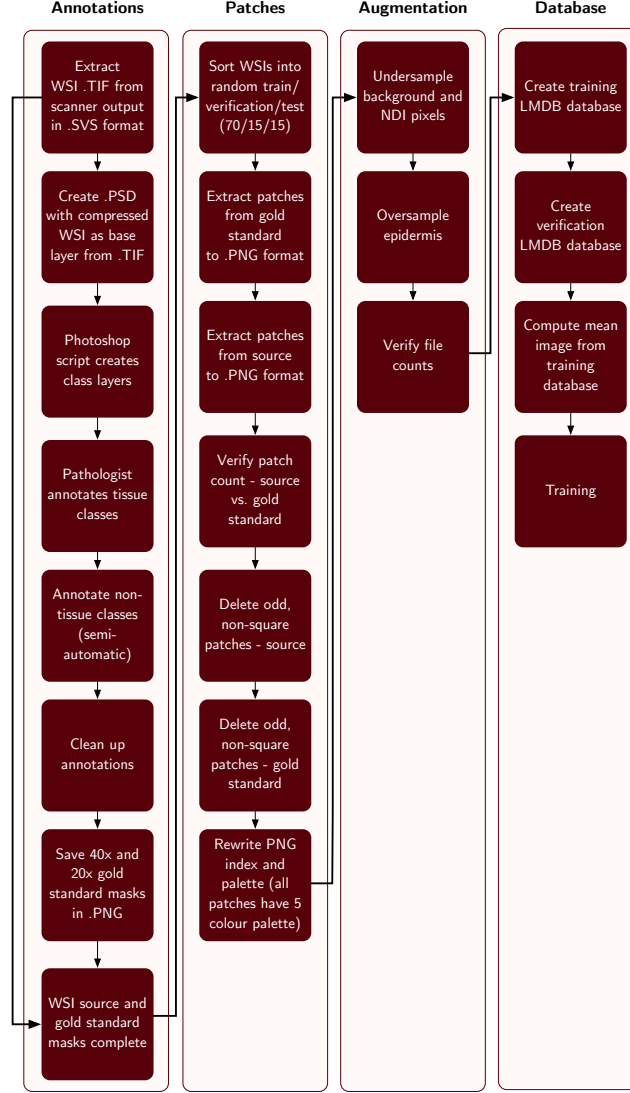


Figure 4.5 – A high-level view of the processing pipeline from source file to ready for training.

4.3 Network Architecture

The network architecture we present is the result of a series of experiments to adapt a previous fully convolutional network (FCN) design by Long et al. [20] for our dataset. Our experiments sought to maximize the simple heuristic of increasing global context for local predictions while constraining GPU memory requirements, network size, and computational complexity. The heuristic is inspired by how a human would examine a slide under the microscope (see Appendix A for a description of the process). Humans

make extensive use of context such as relative size, relative location, probability of objects being in some “scenes” but not in others, when detecting objects and comprehending scenes [75]. This is also the case in semantic segmentation networks as shown by Mostajabi et al. [76] and Liu et al. [77], who demonstrated great improvements on standard datasets by providing global and local context to their networks.

The FCN architecture has become the state-of-the-art approach for semantic segmentation in natural images [78]. This approach trains the network end-to-end to obtain dense pixel-wise predictions. Inputs to FCNs are often whole images, though we use patches due to the gigapixel scale of our source images which well exceed current GPU memory capacity; this reduces the global information available especially in context of the overall slide.

The proposed network uses an FCN architecture with a structure based on the work of Long et al. [20] which itself is based on VGG16 [79] and was converted to an FCN by Long et al. The base structure uses a skip net which combines coarse predictions from deep layers with fine predictions from shallow layers to provide improved overall granularity in the output segmentation. The FCN-16s for example, fuses predictions from the top layer of the stack with predictions from the fourth pooling layer at stride 16. The result is upsampled in a deconvolutional layer to the original input size. The fusion operation is implemented as an element-wise summation of scores from respective streams.

Our work explores two major changes to the base network structure (FCN-16s) to improve the context available for dense prediction. One approach aims to include features at multiple levels of granularity via deconvolution using multiple pixel strides. Using the FCN-16s style network, three levels of output granularity can be obtained based on three architectural variants in three discrete networks. These three network configurations range from a single stream with a 32 pixel stride (FCN-32s) to three streams fused into an 8 pixel stride (FCN-8s). In FCN-16s, scores from each stream are fused (see Figure 4.6) in a step-wise fashion so that intermediate scores from stream 1 and stream 2 are combined and upsampled using deconvolution, then that result is fused before being upsampled with a stride of 16 resulting in the final output score. This combination of layers facilitates local predictions while retaining some global context.

While FCN-16s includes information from multiple streams the final output granularity is fixed. For our segmentation problem we would like to perform prediction based on varying

levels of granularity depending on factors like boundary contour complexity, feature size and patch level homogeneity. To incorporate information at multiple levels of prediction granularity we compute three levels using deconvolution at three independent pixels strides (32,16,8). We then combine these scores using a weighted, element-wise summation, where per-stride weights are hyper-parameters for our model (see Figure 4.7).

The second approach aims to increase the size of the receptive field without changing the filter sizes and thus increasing the compute load. In the base network the receptive field is fixed at 404×404 pixels for all streams at the fc7 convolutional layer. This may be restricting important contextual information that exists at a larger field of view.

To address this challenge, and again taking inspiration from the human approach, there may be features that are easier to detect at different scales. This is analogous to a pathologist changing the objective magnification to see more or less detail. We mimic this process by incorporating multi-scale information into the network by downsampling the higher magnification by a factor of two. This effectively creates two networks learning features at two magnifications, the output of which is upsampled to the higher resolution where the two outputs are fused. In this case the 404×404 pixel receptive field covers a larger area of the input, potentially capturing important spatial information. The downside of this multi-scale approach is a significantly increased memory footprint on the GPU as the core network stack is effectively duplicated (see Figure 4.8).

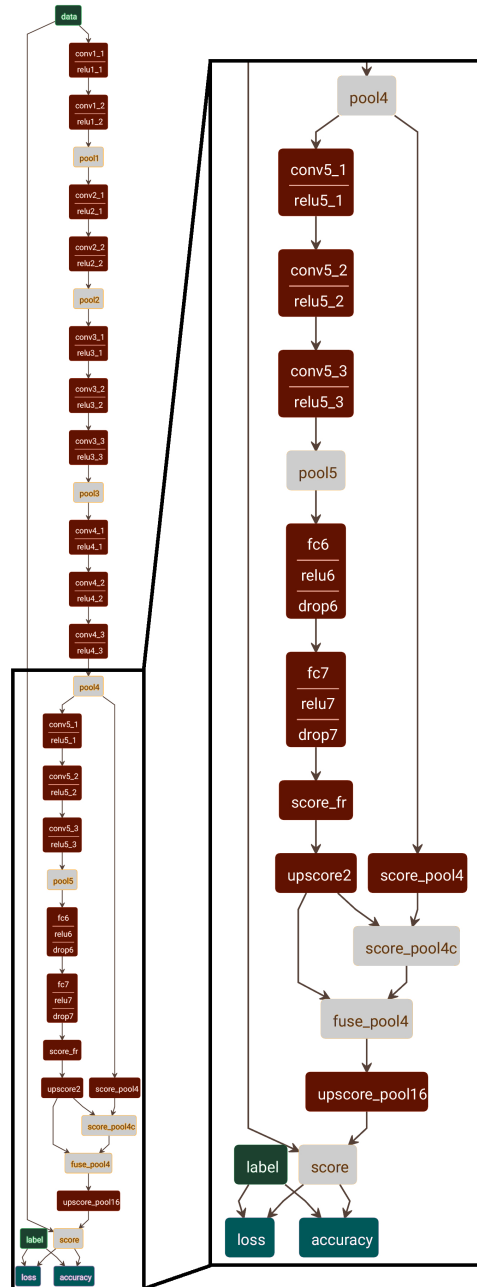


Figure 4.6 – A graphical overview of the FCN-16s architecture. The zoom window shows the fusion of the streams.

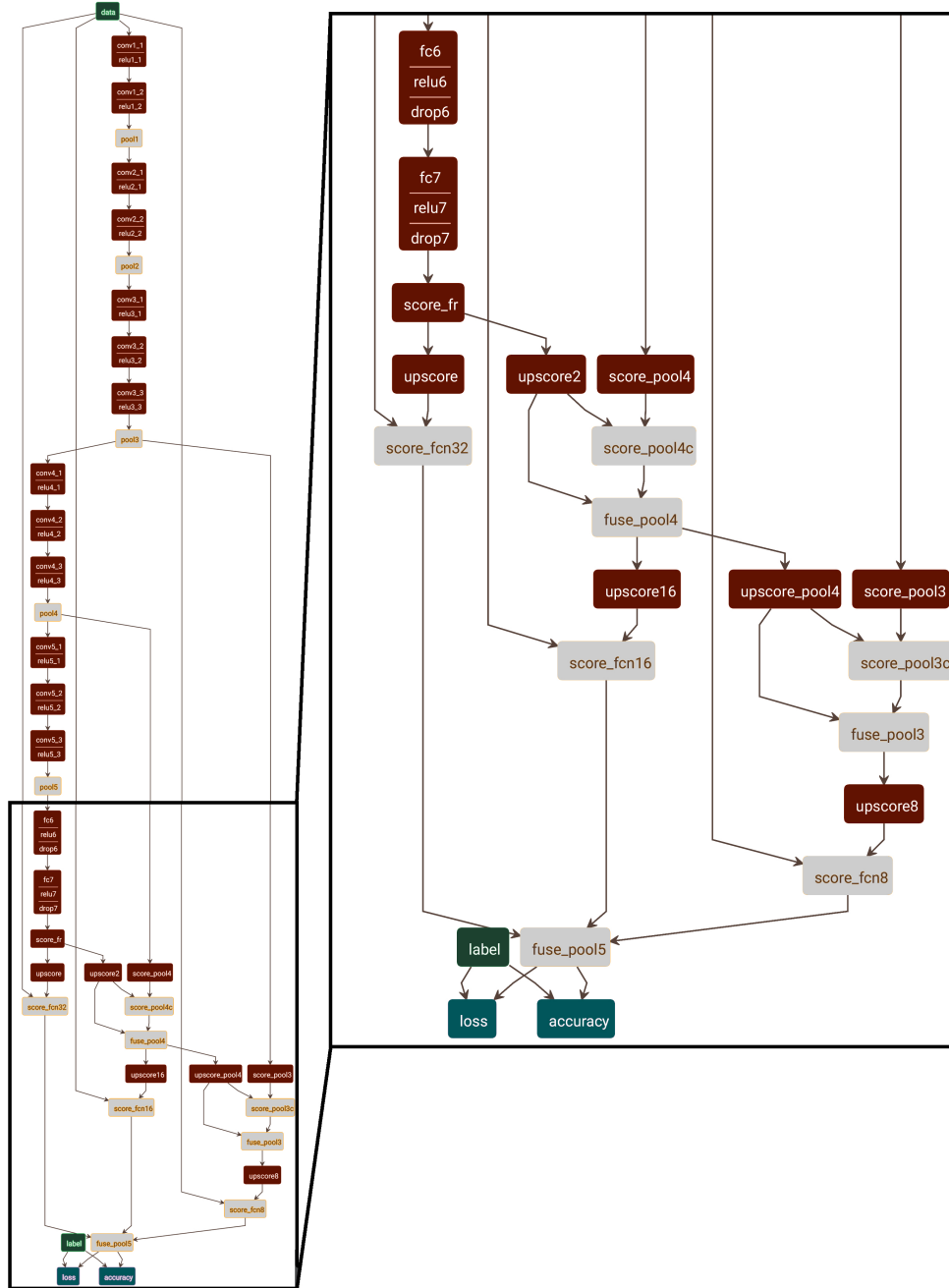


Figure 4.7 – A graphical overview our multi-stride fusion architecture. The zoom window shows the fusion of the multiple strides.

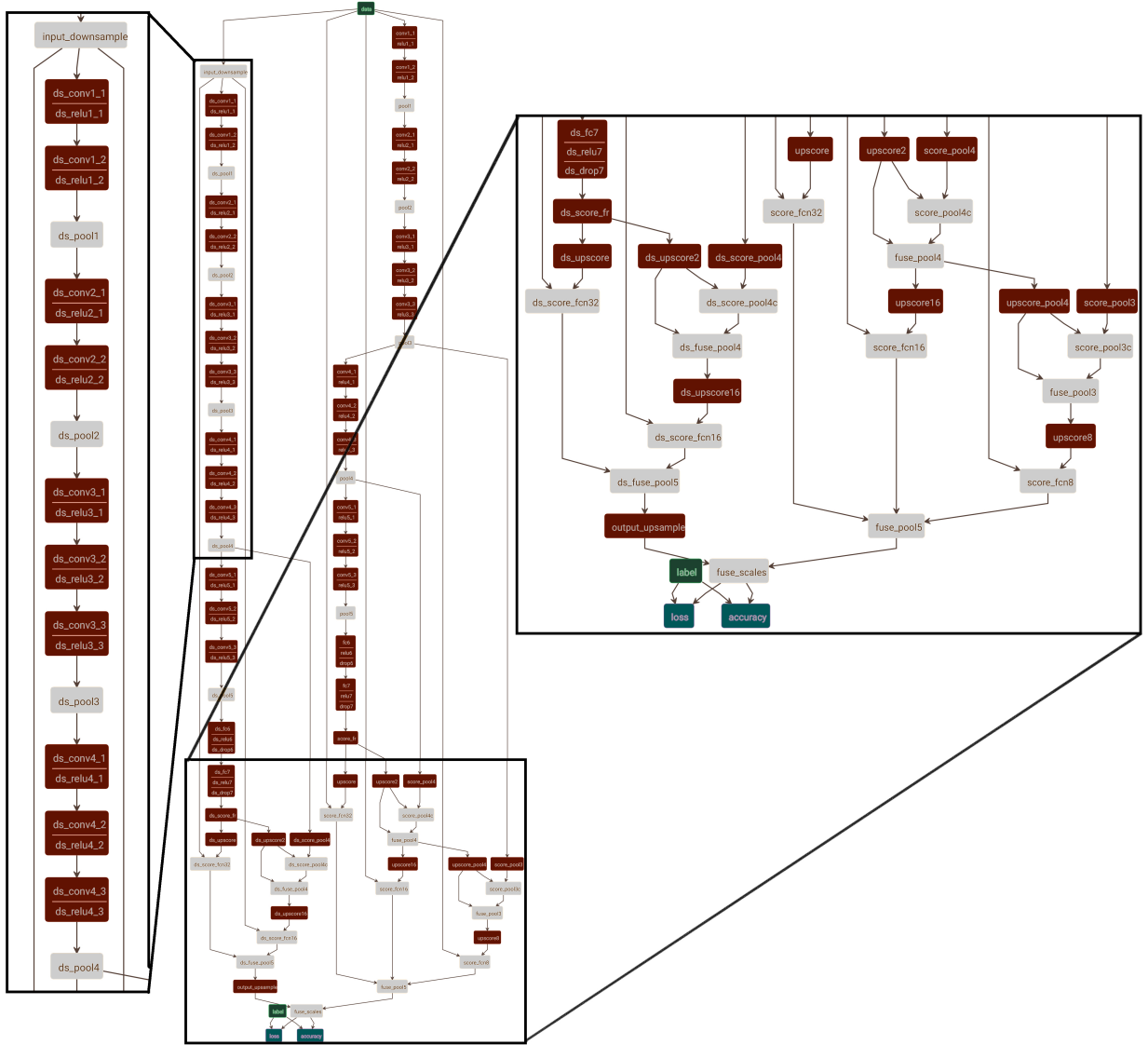


Figure 4.8 – A graphical overview of the multi-stride architecture. The zoom window on the left shows the detail of the downsampled stream. The zoom window on the right shows the fusion of strides from progressively earlier pooling layers, and the required deconvolutions to arrive at the proper output resolution.

4.4 Evaluation Metrics

In order to evaluate the performance of our proposed architecture on our curated dataset we have selected a set of commonly used metrics [80], designed to measure the performance of semantic segmentation inference results.

The metrics we have selected are based on a set of three primary measures of the quality of a segmentation result:

- True positive (TP): correct positive inference
- False positive (FP): incorrect positive inference
- False negative (FN): incorrect negative inference

We use the following notation to describe the various elements of the metrics:

- n_{cl} : The number of classes
- t_i : The total number of pixels in class i
- n_{ij} : the number of pixels of class i predicted to belong to class j .

So for class i and j :

- n_{ii} : The number of correctly classified pixels (TP) precision
- n_{ij} : The number of pixels wrongly classified (FP)
- n_{ji} : The number of pixels wrongly not classified (FN)

Using the above measures we compute the following metrics for our semantic segmentation results:

Pixel Accuracy (PA): Pixel accuracy is a simple metric that represents the ratio of true positive pixels to the sum of all annotated pixels:

$$\text{Pixel Accuracy} = \frac{\sum_i n_{ii}}{\sum_i t_i} \quad (4.3)$$

Mean Pixel Accuracy: Mean pixel accuracy is similar to PA but computes the ratio of true positives class-by-class then averages the result of the total number of classes:

$$\text{Mean Pixel Accuracy} = \frac{1}{n_{cl}} \frac{\sum_i n_{ii}}{t_i} \quad (4.4)$$

Mean Intersection Over Union (mIoU): The intersection over union (IoU) is a standard benchmark for segmentation tasks made popular due to its use in the PASCAL VOC challenges [49]. For our multi-class task we use the meanIoU that first computes the base IoU for each class, followed by the mean over all of the semantic classes:

$$\text{Mean Intersection Over Union} = \frac{1}{n_{cl}} \sum_i \frac{n_{ii}}{(t_i + \sum_j n_{ji} - n_{ii})} \quad (4.5)$$

This represents the degree of overlap between prediction and gold-standard.

Frequency Weighted Intersection Over Union (fwIoU): Frequency weighted intersection over union is an improved version of mIoU where each class is weighted depending on the frequency of their appearance.

$$\text{Frequency Weighted Intersection Over Union} = \frac{1}{\sum_k t_k} \frac{\sum_i t_i n_{ii}}{(t_i + \sum_j n_{ji} - n_{ii})} \quad (4.6)$$

“Final Score” (Score): Inspired by Zhou et al. [81], we compute a “Final Score”. We use the mean pixel accuracy and the mean intersection over union as follows:

$$\text{Final Score} = \frac{\text{Mean Pixel Accuracy}}{\text{Mean Intersection over Union}} = \frac{\frac{1}{n_{cl}} \frac{\sum_i n_{ii}}{t_i}}{\frac{1}{n_{cl}} \sum_i \frac{n_{ii}}{(t_i + \sum_j n_{ji} - n_{ii})}} \quad (4.7)$$

We treat the “Final Score” as the definitive metric for our evaluation.

4.4.1 Evaluation Procedure

While our test slides begin as WSIs, we must send patches to the network for inference. Therefore we generate 512×512 patches for each test case and calculate the selected evaluation metrics patch-by-patch. To obtain the scores for the whole slide test image we first calculate each metric for each patch, then compute the mean scores across all the patches.

To obtain the final evaluation results, we follow the above procedure for each network. For example, for each of the seven test slides we compute the mean patch-wise score, then we compute the mean score across the test slide to get the final score for the network. This process distills over 300,000 data points into the final 4×6 results table.

4.5 Summary

In this chapter we presented the major technical components which together provide a complete description of the work conducted in this thesis. We have described the process we used to curate our dataset, including steps we used to acquire, select and filter cases. We have described the details of our annotation strategy and we covered the tools that we employed to facilitate the efficient use of our annotator’s time and details of their use. We provided details of our augmentation strategy within the context of other work done in this area, which was of critical importance to the process. Finally we described the metrics selected for evaluation of network, with an overview of the procedure.

Experiments and Results

In this chapter we will describe the experiments that were conducted to determine the major technical components of this thesis. While we conducted hundreds of mini-experiments throughout the development process, we only include descriptions for those experiments with a major impact. We separate the experiments into two major areas: Dataset and Network Selection. In the dataset section we describe the process used to inform choices such as patch size. In the network section, we describe the training of our chosen network comparators as well as our two proposed network designs. We present the results of this comparison and discuss our preferred network design. Finally we present the final evaluation of our preferred design including qualitative results where we include the inference of a selection of slides from the test dataset.

5.1 Training Procedure

Experimental models were trained over 20 epochs on the same dataset. For additional training to produce the final results presented in Chapter 5, we initialized the network using pretrained weights from the initial 20 epoch training. From there we trained an additional 20 epoch round on the final training set. It would be more efficient to conduct all of the training in a single monolithic round (and might require fewer overall epochs), however this would likely take over one week to complete. Doing so would occupy the available compute resources in a block that was not practical for our environment.

Before training and inference, we computed the mean image from the respective datasets then normalized the inputs by subtracting the dataset mean image from each. This process

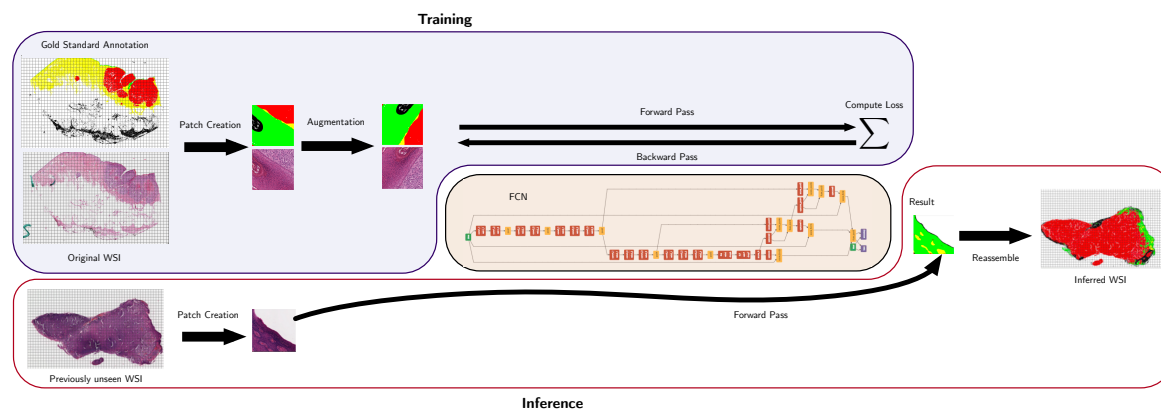


Figure 5.1 – A high level overview of the training and inference stages. The diagram shows two distinct flows through the network. First the training stage, where the patch creation step is highlighted showing the original WSIs and the matching gold standard annotations are divided into non-overlapping patches. Second, in the inference stage, we illustrate the path through the network for previously unseen images that results in the patchwise predictions that are assembled into a whole slide segmentation mask.

was intended to shift the inputs so that the average was closer to zero. This has been shown to improve convergence times [82].

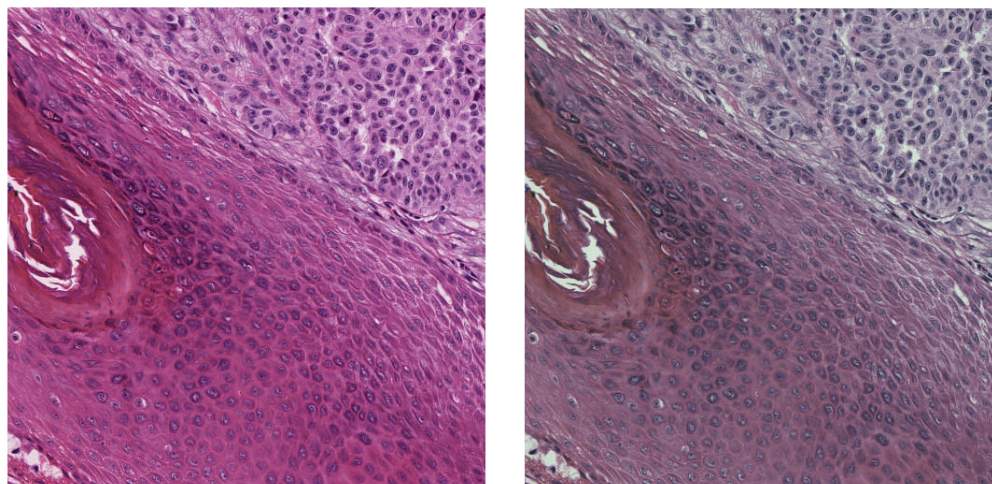


Figure 5.2 – An example result (right image) after subtracting the dataset mean image from the source file.

All experiments and subsequent results were conducted using a single NVIDIA GTX 1080ti GPU. Models were trained using the Caffe open source deep learning framework [83]. We used the “Lightning Memory-Mapped Database” (LMDB) to store dataset images. We also used NVIDIA DIGITS for some training experiments and visualizations.

Hyperparameters

We trained our networks using standard stochastic gradient descent. For most experiments we used a learning rate of 0.0001 with a sigmoid decay pattern to 0.00001. Experimental networks were trained on a subset of available patches (16,000 versus 26,000 in the full training set). To reduce overfitting we implemented a split dropout strategy using a rate of 0.9 at fc6 and 0.75 at fc7.

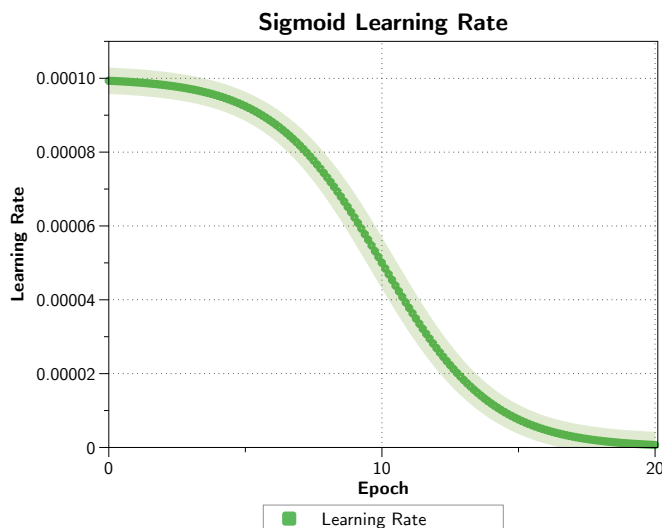


Figure 5.3 – A visualization of the learning rate decay pattern

We used a minibatch of 2 across all networks. Despite claims by Long et al. [20], indicating a minibatch size of 1 is optimal we found no difference between size 1 and 2.

Transfer learning

Notwithstanding the significant difference in domain specific features between the PascalVOC dataset and our medical images, transfer learning was essential to improving our training times and results. We made extensive use of transfer learning to reduce training time and improve results. We initialized weights for learnable layers using publicly available models (FCN-32s, FCN-16s, FCN-8s) and fine-tuned from there. We used net-surgery to adapt existing models to our architecture when using multiple streams and layers are duplicated but names must change (meaning pretrained weights will not be applied automatically). These pretrained model architectures generally match our own up to the fc7 convolutional layer. For our multi-stride networks we initialized weights using multiple pretrained networks trained at each requisite stride for the given branch of our network.

Non-matching weights were learned from random initializations.

Loss function

To compute training loss we used the softmax (Equation 5.1) with multinomial loss (Equation 5.2):

$$p(y^{(i)} = k | x^{(i)}; \theta) = \frac{e^{\theta_k^T x^{(i)}}}{\sum_{k=1}^K e^{\theta_k^T x^{(i)}}} \quad (5.1)$$

$$J(\theta) = \sum_{i=1}^m \sum_{k=1}^K 1\{y^{(i)} = k\} \log p(y^{(i)} = k | x^{(i)}; \theta) \quad (5.2)$$

where m is the sample and K is the class.

5.2 Network Selection Results

We trained four distinct networks, two networks that incorporated our multi-stride design and two additional, previously presented networks for comparison. The purpose of these experiments was to evaluate our designs against a competent baseline network in order to decide which of our two designs to proceed with for additional training and generation of final inference results.

We trained these networks using 16,675 patches from the training set with a minibatch size of 2. We trained each for 20 epochs (approximately 167,000 iterations). All networks were fine-tuned using publicly available pretrained weights. Note that we did attempt training from scratch using FCN-16s with limited success and opted for fine-tuning as a time saving and performance improving measure.

For each network, we present inference results based on the metrics described in Chapter 4, Section 4.4 as conducted on each of our test slides. In Figure 5.4 through Figure 5.7 we include the training performance plots to show that training proceeded in a similar way for each network and that the hyperparameters used were reasonable (not perfect) for each network. We interpret these plots according to the standard heuristics as follows:

- Learning rate: from the validation loss curve (red) we see the curve drops rapidly at first then settles into a smooth curve. We could have lowered the learning rate more.
- Batch size: from the training loss curve (blue), we see that there is some “noise” due to the variance between batches (we used a batch size of 2). This could be reduced by increasing the batch size, but it appears under control. Learning rate could be lowered.
- Overfitting: Looking at the validation loss curve (red) and the training loss curve (blue), we see they are very close, perhaps training loss is slightly lower which could indicate very mild overfitting.

The results were conducted on the test dataset consisting of seven whole slide images of various sizes and class composition. To provide some context we summarize the image properties in Table 5.1.

Table 5.1 – Image details for our whole slide test cases.

Case ID	Size (MB)	Pixel Size	# of Patches
BF-A5EO	536.5	15438×11582	660
D9-A3Z4	1146.1	23086×16546	1440
EB-A3XE	1098.1	21420×17087	1353
EE-A2GO	1407.4	24751×18952	1776
EE-A182	1761.8	27845×21088	2214
ER-A19H	2276.1	30463×24903	2832
WE-A8K4	1937.4	29381×21978	2394

We also provide a qualitative comparison in Figure 5.9.

- **FCN-AlexNet**

We include the AlexNet architecture for performance and complexity comparison. We converted the original network containing standard fully connected layers to a fully convolutional model using the procedure described by Long et al. [20].

As we expected FCN-AlexNet’s overall score compared to the other networks was lower. We included FCN-AlexNet to demonstrate the performance of a popular network on our dataset. These results also demonstrate weaknesses in the standard metrics. For example, as shown in Figure 5.4 and Figure 5.5, FCN-AlexNet outperforms the baseline network FCN-16s in five out of seven cases on pixel accuracy. However, looking at Figure 5.9 we can see that FCN-16s clearly shows better qualitative performance.

FCN-AlexNet by virtue of its coarse segmentation, covering areas with the correct label for the majority class even though the area contains annotations from multiple classes at finer levels of detail.

Please refer to Table 5.2 and Figure 5.4 for training results from this network.

- **FCN-16s (Baseline Network)** The baseline network is a common reference and a much used design for learning applications in medical imaging (see Related Work). This network had posted state-of-the-art results when released by Long et al. in 2015, and continues to be used as a standard network design for medical segmentation tasks [84].

This network as the baseline, performs well on areas where tissues exhibit features that are truly representative of that tissue class. When features become more poorly differentiated FCN-16s struggles. For example in slide “F” in Figure 5.9, the dermal areas (marked in yellow) shown in the gold standard are almost completely missed. In the “Original” tissue source, these areas are stained much darker than usual causing (we think) the network to rely more on finer structural details to make the correct inference.

Please refer to Table 5.3 and Figure 5.5 for training results from this network.

Note that we also tested a baseline two-stream FCN-16s network as a control for scale, but decided not to include the results in our final analysis. Please see Appendix E for results from this network.

- **Multi-Stride Fusion**

Our multi-stride method increased context available to the network by performing an element-wise fusion of the deconvolution using three distinct stride values from three streams connected via skip connections to progressively higher levels of the network. This effectively combines three distinct networks, each at a different granularity of inference. The intuition with this design was that the network would be more effective at capturing the spectrum of features between coarse and fine.

Our multi-stride fusion outperformed the baseline score by 7 percentage points and FCN-AlexNet by 9 percentage points with the same data and training conditions. As

shown in Figure 5.9, the network’s qualitative performance is much better in areas where finer structural details seem to be more important than shape and colour information. As discussed above in Table 5.2, slide “F” in Figure 5.9 is a plausible example where this is important for the correct inference of the dermal areas (marked in yellow).

Please refer to Table 5.4 and Figure 5.6 for training results from this network.

- **Multi-Stride, Multi-Scale Fusion**

Here we took our multi-stride fusion design and incorporated an additional stream. Where the multi-stride segment operates on images at 40x objective magnification, the multi-scale segment downsamples the input to a 20x equivalent magnification. As discussed in Chapter 2 (Related Work), there are many examples in the literature using some form of multi-scale information. In our multi-scale segment we do not require the training of multiple networks separately or in an ensemble; instead we downsample on-the-fly. Since the downsampled images contain significantly less information, instead of using a standard downsampling algorithm (e.g. bilinear, bicubic) we downsample using max-pooling due to the significant performance cost with other methods.

Our multi-stride, multi-scale fusion design outperforms all other networks tested, including our multi-stride fusion network. Somewhat unexpectedly the performance improvement over multi-stride was less than one percent, but with a much higher computational cost and memory footprint (see Figure 5.7). The overall inference time for the test set was 48 seconds slower. One of the goals of our network design was to keep the computational demand as close to the baseline as possible. It was also important to keep the training as simple as possible by not merging two networks with discrete training operations. It seems likely that the use of max-pooling based downsampling, which addresses these design goals, is also less effective than standard downsampling would be.

Please refer to Table 5.5 and Figure 5.7 for training results.

Table 5.2 – Inference results for FCN-AlexNet

FCN-AlexNet						
Case ID	PA	mPA	mIoU	fwIoU	Score	Time(s)
BF-A5EO	0.94109	0.54387	0.51733	0.91216	0.53060	18.88026
D9-A3Z4	0.88253	0.49888	0.45304	0.81527	0.47596	40.84072
EB-A3XE	0.88207	0.51848	0.47070	0.82806	0.49459	38.33455
EE-A2GO	0.87177	0.50599	0.46932	0.83185	0.48765	50.34736
EE-A182	0.89934	0.51584	0.48265	0.87756	0.49924	62.81422
ER-A19H	0.86111	0.59890	0.53659	0.80172	0.56775	80.29802
WE-A8K4	0.89493	0.52769	0.50927	0.87958	0.51848	67.88112
Mean:	0.89041	0.52995	0.49127	0.84946	0.51061	51.34232

Table 5.3 – Inference results for FCN-16s

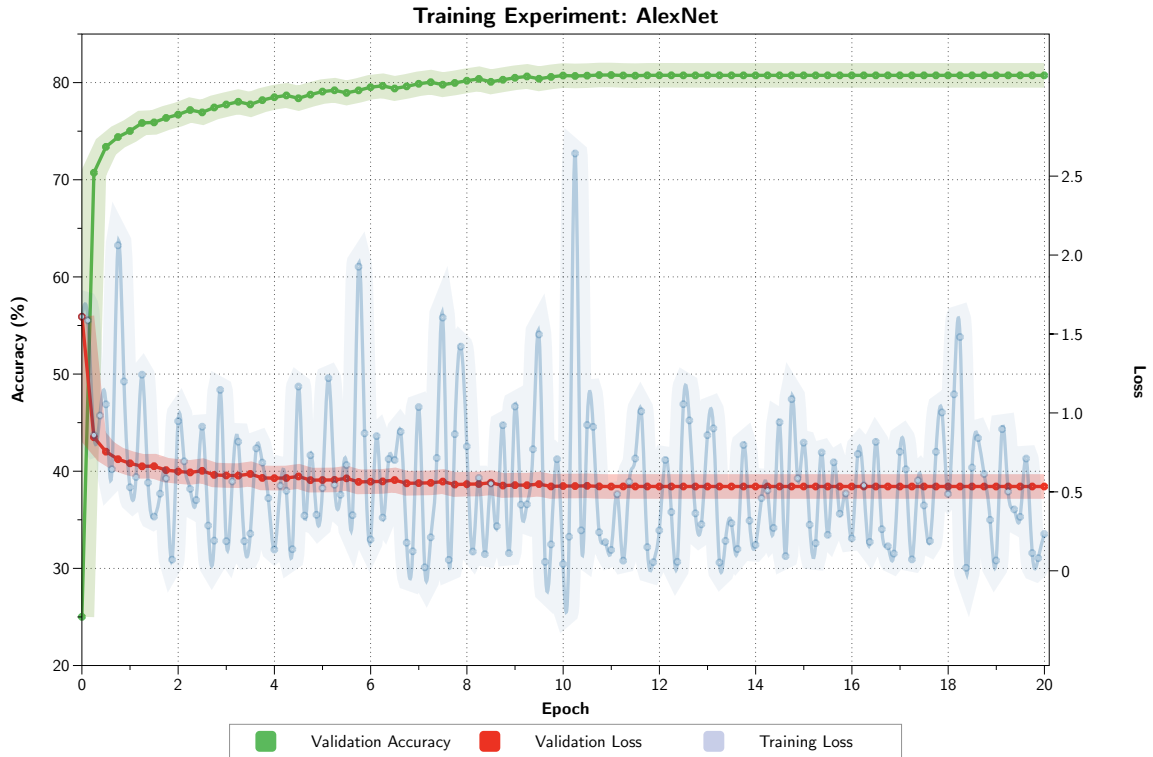
FCN-16s						
Case ID	PA	mPA	mIoU	fwIoU	Score	Time(s)
BF-A5EO	0.9394	0.5795	0.5399	0.9149	0.5597	51.0955
D9-A3Z4	0.8735	0.5231	0.4758	0.8172	0.4995	110.8822
EB-A3XE	0.8834	0.5375	0.4908	0.8347	0.5141	104.1202
EE-A2GO	0.8778	0.5318	0.4910	0.8434	0.5114	136.9015
EE-A182	0.8855	0.5424	0.4929	0.8677	0.5176	170.7600
ER-A19H	0.8511	0.6335	0.5686	0.8060	0.6010	218.3083
WE-A8K4	0.8322	0.5114	0.4888	0.8193	0.5001	184.4339
Mean:	0.8775	0.5513	0.5068	0.8433	0.5291	139.5002

Table 5.4 – Inference results for Multi-Stride Fusion

Multi-Stride						
Case ID	PA	mPA	mIoU	fwIoU	Score	Time(s)
BF-A5EO	0.9640	0.6887	0.6454	0.9479	0.6670	48.4411
D9-A3Z4	0.9119	0.5976	0.5516	0.8635	0.5746	105.7572
EB-A3XE	0.9221	0.6165	0.5715	0.8810	0.5940	100.3250
EE-A2GO	0.9148	0.6176	0.5749	0.8844	0.5963	131.6688
EE-A182	0.8724	0.6180	0.5563	0.8598	0.5872	164.6194
ER-A19H	0.8775	0.7056	0.6496	0.8435	0.6776	210.1503
WE-A8K4	0.7742	0.5243	0.4975	0.7638	0.5109	177.4948
Mean:	0.8910	0.6240	0.5781	0.8634	0.6011	134.0652

Table 5.5 – Inference results for Multi-Stride + Multi-Scale Fusion**Multi-Stride + Multi-Scale**

Case ID	PA	mPA	mIoU	fwIoU	Score	Time(s)
BF-A5EO	0.9639	0.6854	0.6433	0.9477	0.6644	64.9084
D9-A3Z4	0.9119	0.6014	0.5541	0.8634	0.5777	136.6677
EB-A3XE	0.9223	0.6149	0.5705	0.8811	0.5927	128.4278
EE-A2GO	0.9165	0.6219	0.5779	0.8862	0.5999	168.3807
EE-A182	0.8871	0.6298	0.5647	0.8742	0.5973	210.6673
ER-A19H	0.8835	0.7102	0.6538	0.8494	0.6820	269.4744
WE-A8K4	0.7680	0.5208	0.4948	0.7578	0.5078	227.9131
Mean:	0.8933	0.6264	0.5799	0.8657	0.6031	172.3485

**Figure 5.4** – Visualization of the training process for the FCN-AlexNet Network

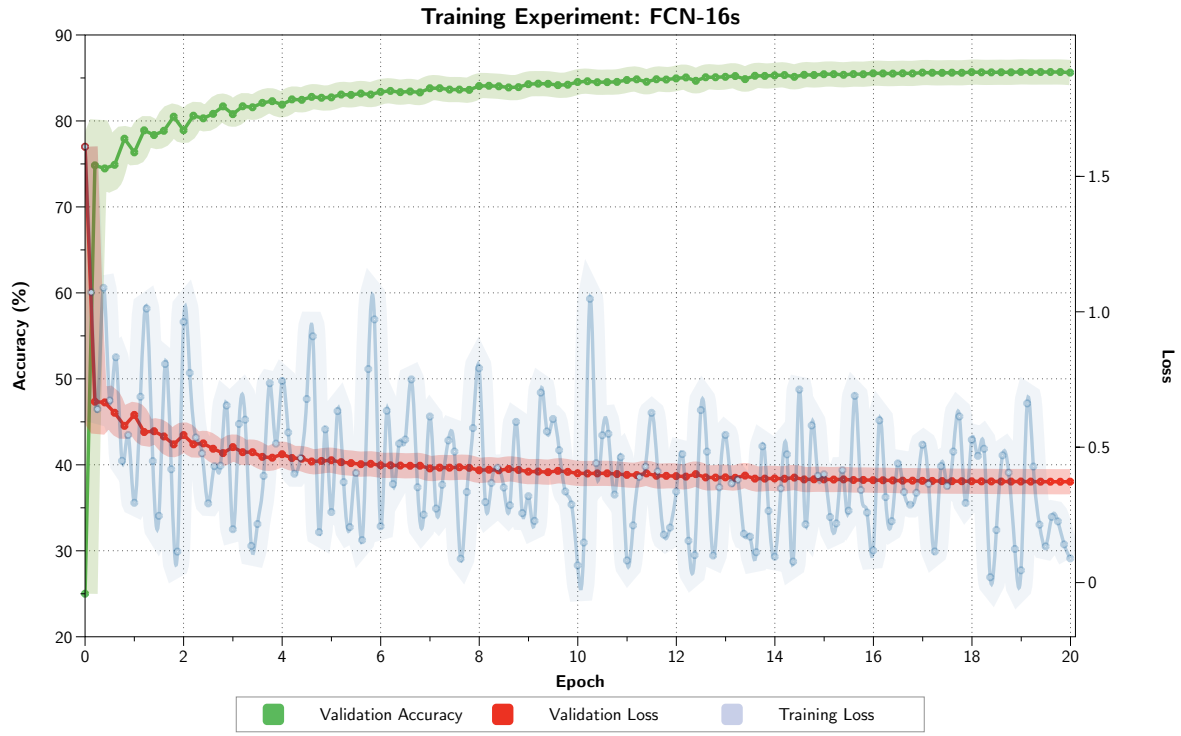


Figure 5.5 – Visualization of the training process for the FCN-16s Network

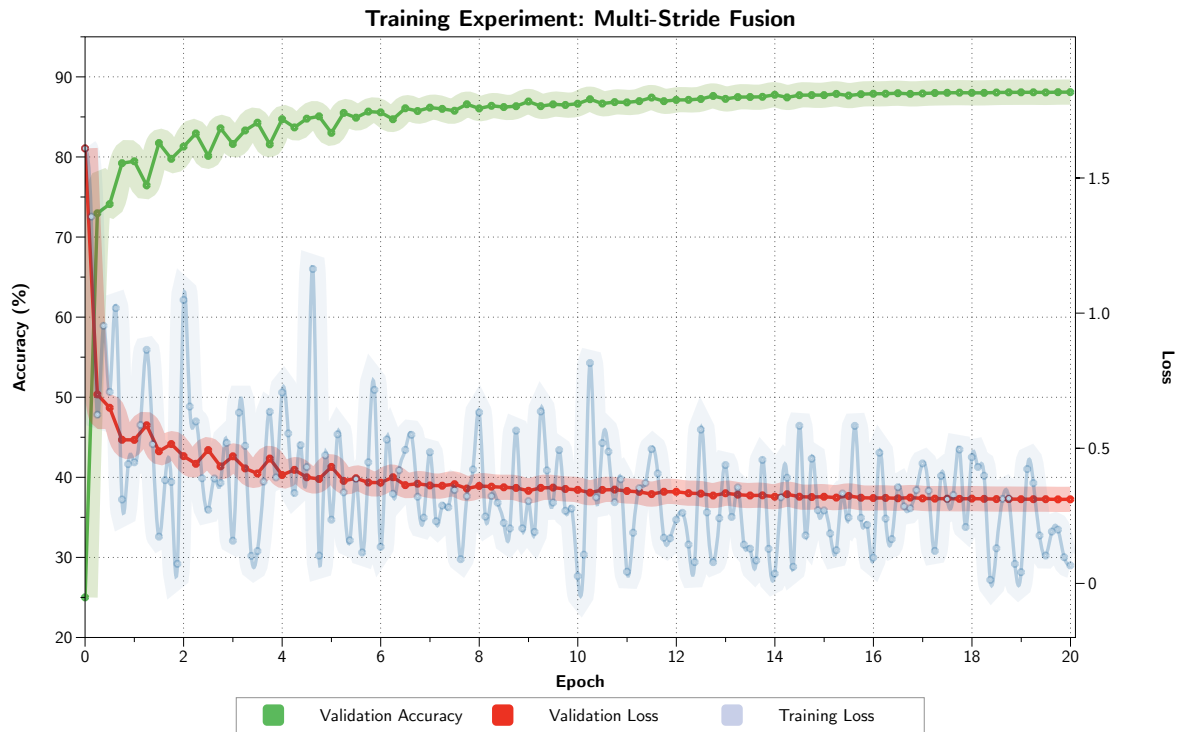


Figure 5.6 – Visualization of the training process for the Multi-Stride Fusion Network

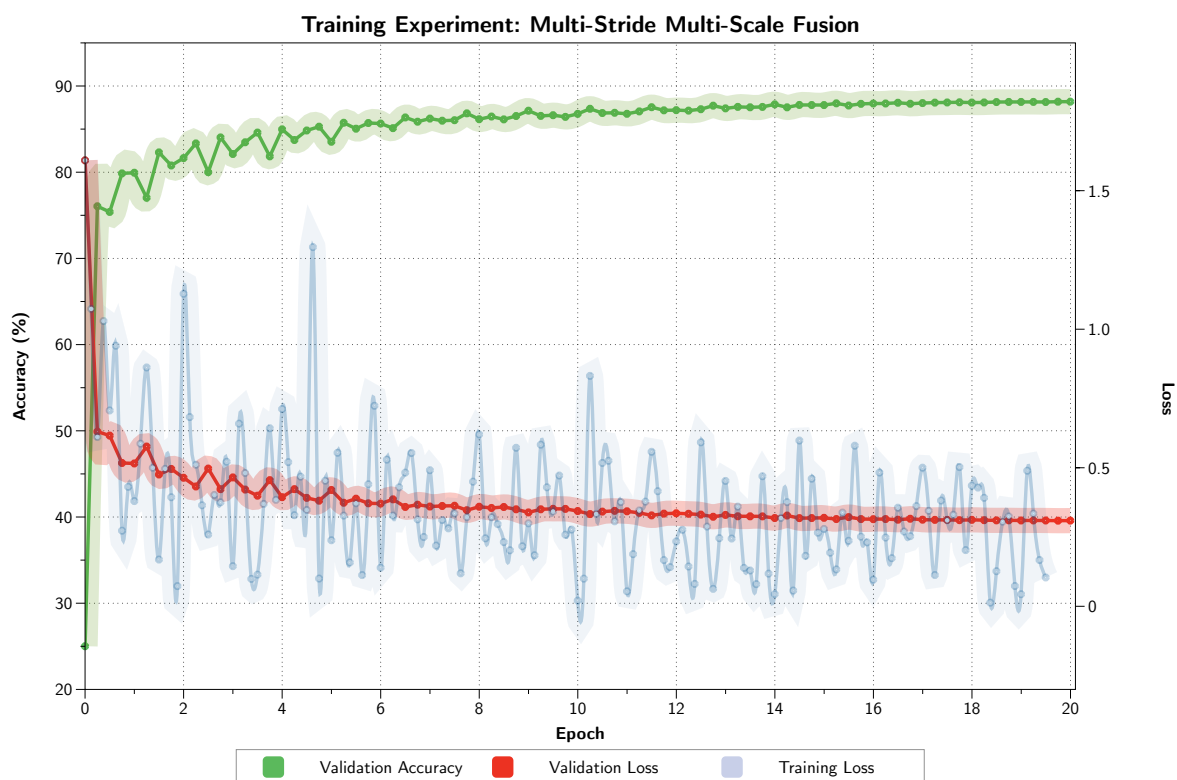


Figure 5.7 – Visualization of the training process for the Multi-Stride, Multi-Scale Fusion Network

5.2.1 Network Selection Results Summary

Table 5.6 – A summary of the network training results showing the mean values computed across each of the test slides for each respective network. Bold text represents the best score, green represents the network selected for further training.

Network	PA	mPA	mIoU	fwIoU	Score	Time (s)
AlexNet	0.8904	0.5300	0.4913	0.8495	0.5106	51.3423
FCN-16s	0.8775	0.5513	0.5068	0.8433	0.5291	139.5002
Multi-Stride	0.8910	0.6240	0.5781	0.8634	0.6011	134.0652
Multi-Stride Multi-Scale	0.8933	0.6264	0.5799	0.8657	0.6031	172.3485

5.2.2 Discussion of Experimental Results

Network Selection

During our initial testing with much smaller datasets, the multi-stride, multi-scale network outperformed the multi-stride alone by nearly two percentage points. However during the inference testing for network selection we saw that this gap dropped to within 0.3% (see Figure 5.8). While in many applications this difference would be significant, for the purposes of this thesis such a performance improvement over the multi-stride alone did not justify the additional computational load imposed by adding the additional stream for scale. Since one of the primary goals of the network design was to optimize the performance-to-compute ratio, we chose to focus on the multi-stride network for further training. Even so, this does not invalidate the multi-stride + multi-scale design, but we reserve further experimentation using this design for future work.

We have described our proposed network designs, however numerous other permutations of these designs were tested. The most notable design element that was not incorporated into the designs presented was the inclusion of additional stride deconvolutions, and associated skip connections. While in the full-scale stream we use three strides we did additional testing on four and five stride configurations. We found minimal improvements in performance despite a large increase in training time.

Inference Times

A somewhat surprising result is in the inference times of the multi-stride network versus

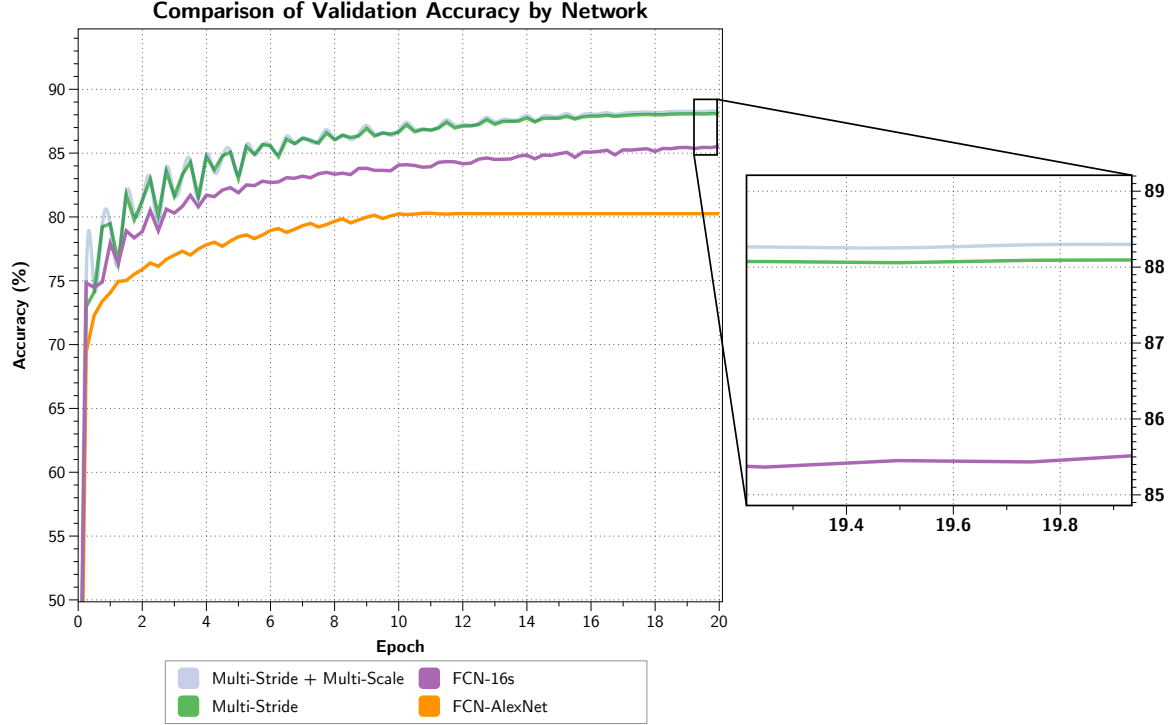


Figure 5.8 – A summary of the validation accuracies generated during the training process. Here we show the results for the four networks together. We observe that the multi-stride, and the multi-stride + multi scale networks achieved a validation accuracy within 0.3% of each other (88.14% and 88.33% respectively).

the baseline FCN-16s network. Since the multi-stride network is a more complex design, one would expect inference to be slower even if only slightly. What we found was that multi-stride was actually faster. From Table 5.6 we see that the mean inference time for a WSI was about five seconds faster. This suggest the multi-stride fusion network was able to learn a less complex function (filter) mapping than the FCN-16s network.

Memory Usage

The networks we tested have a memory use footprint proportional to their size (i.e. number of layers). The peak memory usage for the candidate networks was as follows:

- AlexNet: 4.21 GB
- FCN-16s: 5.73 GB
- Multi-Stride: 6.57 GB
- Multi-Stride, Multi-Scale: 9.98 GB

These results are based on a minibatch of size 2. Multi-Stride fits comfortably within the 8GB available on most consumer GPUs being used for learning applications. Multi-Stride, Multi-Scale has a large memory footprint, requiring at least 10GB. This arises from the total of five-strides in this network. This could be partially mitigated by using a single stride (instead of two) in the downsampled stream combined with a minibatch size of 1.

Hyperparameters

We spent significant time testing hyperparameters for the baseline FCN-16s network. Most of this testing was done at the proof-of-concept stage to firmly establish that our research questions were tractable. We focused on the following hyperparameters which transferred well to the other network designs:

- Learning Rate
- Rate Decay
- Minibatch size
- Dropout
- Training epochs
- Momentum
- L^2 weight decay

We manually tuned these hyperparameters using a guess and check approach while referencing Bengio’s guide [85]. This processes resulted in the following settings:

Table 5.7 – Settings used for primary training hyperparameters

Hyperparameter	Setting
Learning rate	0.0001
Rate decay	Sigmoid to 0.00001
Minibatch size	2
Dropout	0.6,0.9
Training epochs	20
Momentum	0.9
L^2 weight decay	0.0000007

It is not certain that these parameters are optimal for the dataset. Other strategies discussed by Bengio [85] could be used in the future to search for improved settings.

5.3 Final Results

Here we present our final results using the selected multi-stride network, trained for a total of 40 epochs, with 20 epochs trained on the full training set (approximately 26,000 patches) initialized with weights from the experimental training result which was trained for 20 epochs. We randomly selected six patches as exemplars for qualitative comparison and we perform a complete inference including tumour detection on four whole slide images (see Figure 5.9). For the whole slide results we trained the network using our standard patch-based method, and reassembled the inferred patches into whole slide segmentation and detection masks (see Figure 5.10).

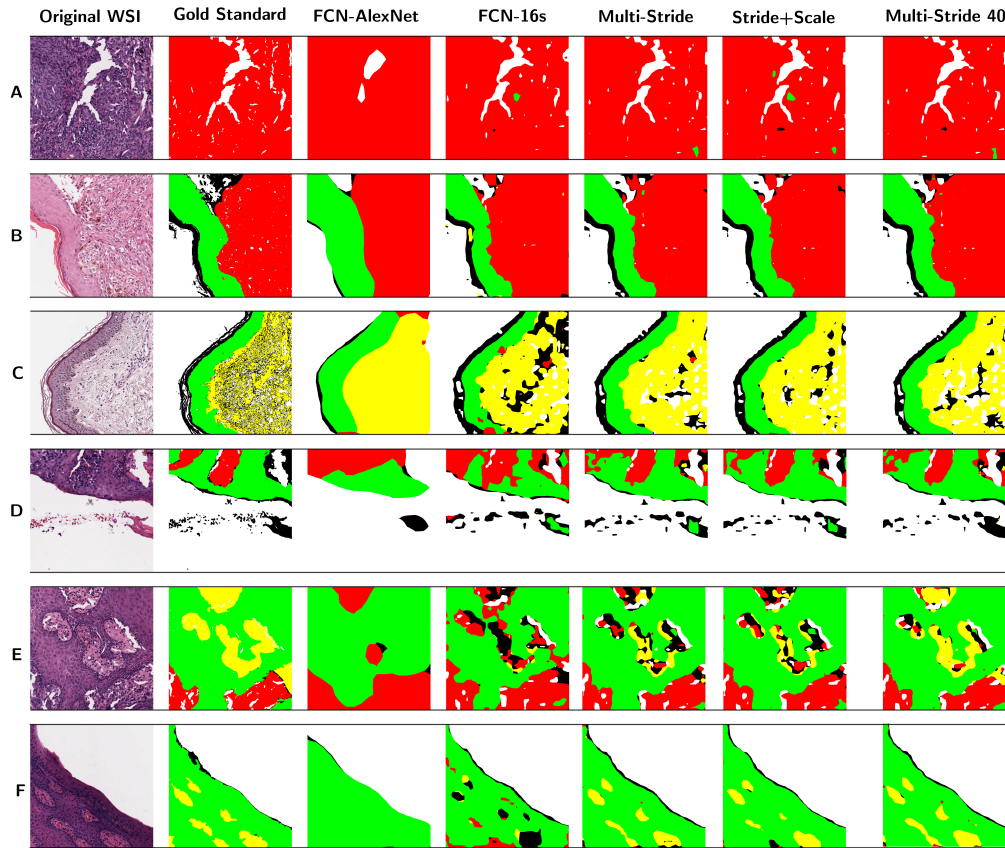


Figure 5.9 – Qualitative comparison of each network. Each network was trained for 20 epochs on the experimental training set except “Multi-Stride 40” which was trained for an additional 20 epochs with an expanded dataset. Scores for each network are available in Table 5.8

. Note the FCN-16s example in the bottom row was unable to identify the dermis regions embedded in the epidermis, while the multi-stride fusion network successfully identifies these regions.

Table 5.8 – Metrics resulting from the inference of the patches presented in Figure 5.9. See Section 4.4 for details on the metrics used here.

	Network	PA	mPA	mIoU	fwIoU	Score	Time(s)
A	AlexNet	0.9243	0.6151	0.5698	0.8611	0.5924	0.1162
	FCN-16s	0.9012	0.6624	0.5758	0.8417	0.6191	0.2776
	Multi-Stride	0.9659	0.8538	0.8152	0.9377	0.8345	0.2729
	Multi-Stride Multi-Scale	0.9625	0.8630	0.8206	0.9356	0.8418	0.5107
	Multi-Stride 40 Epochs	0.9663	0.8578	0.8207	0.9391	0.8393	0.2708
	Network	PA	mPA	mIoU	fwIoU	Score	Time(s)
B	AlexNet	0.6266	0.6120	0.4235	0.4494	0.5177	0.0283
	FCN-16s	0.5750	0.5608	0.4127	0.4307	0.4868	0.0683
	Multi-Stride	0.7263	0.7189	0.5885	0.5878	0.6537	0.0697
	Multi-Stride Multi-Scale	0.7255	0.7183	0.5875	0.5882	0.6529	0.0916
	Multi-Stride 40 Epochs	0.7289	0.7192	0.5894	0.5886	0.6543	0.0669
	Network	PA	mPA	mIoU	fwIoU	Score	Time(s)
C	AlexNet	0.8464	0.6840	0.5841	0.7332	0.6340	0.0284
	FCN-16s	0.8543	0.7238	0.6439	0.7499	0.6839	0.0688
	Multi-Stride	0.9189	0.8384	0.7809	0.8500	0.8096	0.0697
	Multi-Stride Multi-Scale	0.9125	0.8216	0.7623	0.8389	0.7920	0.0913
	Multi-Stride 40 Epochs	0.9239	0.8477	0.7922	0.8585	0.8200	0.0668
	Network	PA	mPA	mIoU	fwIoU	Score	Time(s)
D	AlexNet	0.6849	0.3658	0.2283	0.5082	0.2971	0.0285
	FCN-16s	0.6561	0.4180	0.2580	0.5465	0.3380	0.0689
	Multi-Stride	0.7981	0.5574	0.4247	0.7225	0.4910	0.0695
	Multi-Stride Multi-Scale	0.8009	0.5544	0.4168	0.7190	0.4856	0.0933
	Multi-Stride 40 Epochs	0.8658	0.6268	0.5026	0.7967	0.5647	0.0759
	Network	PA	mPA	mIoU	fwIoU	Score	Time(s)
E	AlexNet	0.7931	0.6121	0.4620	0.6734	0.5371	0.0286
	FCN-16s	0.7914	0.6693	0.4895	0.6927	0.5794	0.0687
	Multi-Stride	0.8940	0.7911	0.6652	0.8279	0.7281	0.0757
	Multi-Stride Multi-Scale	0.9057	0.8109	0.6905	0.8416	0.7507	0.0921
	Multi-Stride 40 Epochs	0.9164	0.8304	0.7199	0.8589	0.7752	0.0742
	Network	PA	mPA	mIoU	fwIoU	Score	Time(s)
F	AlexNet	0.9047	0.5145	0.4665	0.8263	0.4905	0.0283
	FCN-16s	0.8835	0.5704	0.4921	0.8489	0.5312	0.0690
	Multi-Stride	0.9558	0.8208	0.7169	0.9280	0.7688	0.0744
	Multi-Stride Multi-Scale	0.9538	0.7694	0.6917	0.9207	0.7305	0.0923
	Multi-Stride 40 Epochs	0.9632	0.8406	0.7544	0.9377	0.7975	0.0745

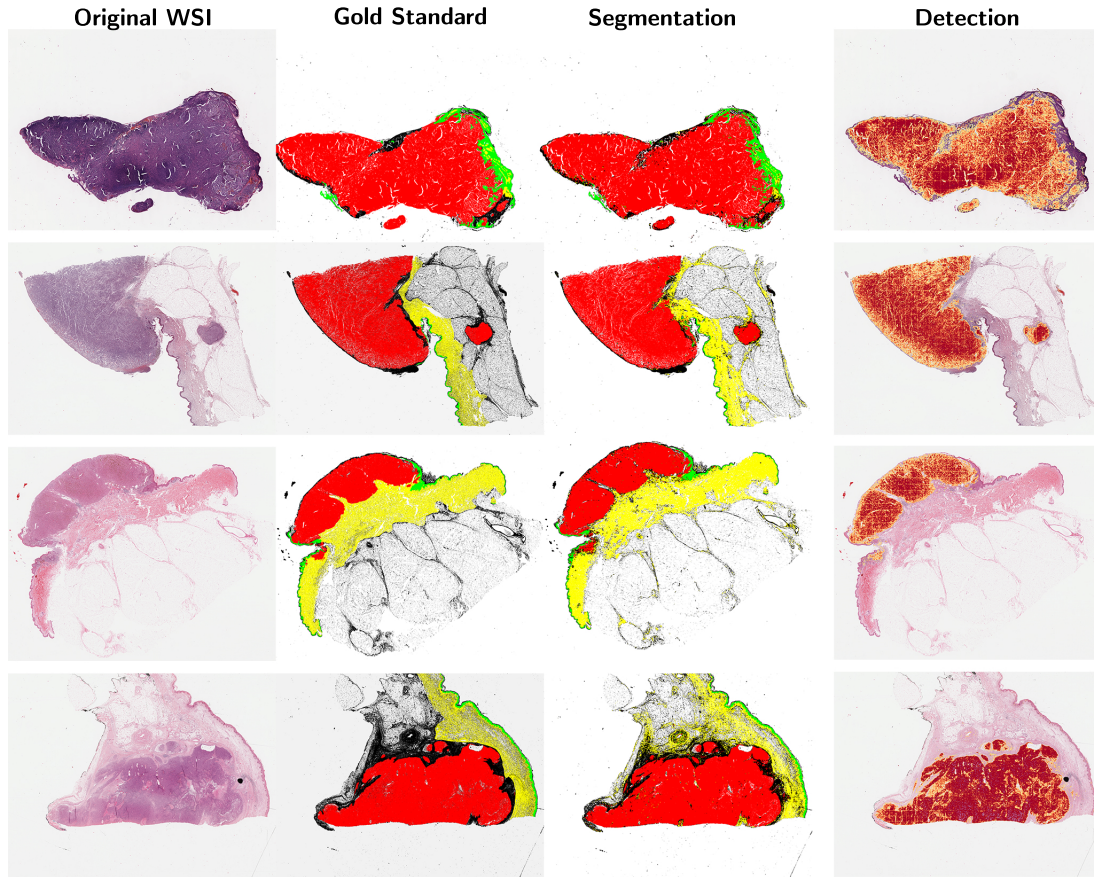


Figure 5.10 – Demonstration of the multi-stride network trained for 40 epochs on four whole slide images from the test set. Detection results show a confidence map with values indicating the confidence that at a given pixel, tumour is present. Darker colours represent a higher likelihood of the pixel containing tumour information.

5.3.1 Discussion of Final Results

Dataset

In most cases in the literature, experiments are conducted on datasets that the researchers had little control over. This obfuscates how important decisions made during the curation of a dataset can be with respect to the quality of the trained model. For example, the automated methods used to reduce annotation time resulted in the data being “hyper-supervised”. That is to say, many individual pixels were labelled in a way that seems deliberate, but would have been better labelled as the surrounding class instead (especially “NDI” class labels).

Dataset Size

The WSIs were annotated in blocks of five. As blocks were completed additional training experiments were completed to determine if performance would improve with additional data. We found that the network was capable of learning starting at approximately 10,000 patches, and we continued to see improvements until the initial pool of fifty slides were annotated. The network selection experiments (see Section 5.2) used approximately 16,000 patches and the final training used 26,000 patches.

We also experimented with the validation set size, testing sizes of 10%, 15%, and 20%. We found that 15% performed the best compared to 10%, and that 20% showed signs of overfitting, and added significant time to the training process.

Data Augmentation

We discuss data augmentation in detail in subsection 4.2.6 of Chapter 4, though based on the results of the WSI inference we realized that our model struggles with the variability of the H&E intensity across patients (see Figure C.1). We believe many of the issues we have encountered could be addressed with additional augmentation, by modulating the intensity of the colours of the patches to mimic the variation found across our dataset.

Patch Size

The source, gigapixel WSIs are much too large to be used as input to the training process due to memory constraints. Therefore we needed to divide the source image into smaller “patches” of a size that could be used with the network for training. The selection of the patch size could have had a substantial effect on performance of the network. The primary goal of our network design is to maximize the context available on global and local scales within the receptive field of our network. In the literature most work that operates on WSIs use patch-level training [14, 74, 51, 86]. Various patch sizes emerged ranging from 64×64 to 2048×2048 and higher in powers of two or some even number divisible by 100. The common feature of most patch sizes is their ability to be divided by two multiple times to naturally maintain integer pixel dimensions after multiple reductions resulting from max-pooling operations or convolutional operations.

To assist in determining an appropriate patch size we trained a basic FCN-16s model on a preliminary sample of our dataset (dataset annotation was still in process) using 256, 512, 1024, and 2048 square pixel patches. We chose the FCN-16s network since the receptive field sizes would most closely match our planned network design. Since the

data was limited, the qualitative performance of these tests was not particularly useful, however the results did indicate slightly improved performance using the 512 pixel patches. Conducting this test also highlighted some other factors important to the selection of the patch size. Firstly smaller patch size results in significantly more files to manage. Managing and processing hundreds of thousands of files adds time and complexity to the workflow. Read/write overhead for individual files adds significant time to processing stages, browsing processing results is not possible using GUI tools and there is network preparation overhead during inference on each patch. Secondly, larger patch sizes use more GPU resources which could be a factor for complex models. Larger patches either reduce available pixels since source images cannot be divided evenly by patch size, this as much as patch-size minus one pixel may be removed from the left and bottom edges or to avoid this an additional padding step must be included to make source images evenly divisible by the patch size. Finally, and most importantly, the additional context available in the larger patch sizes can only be taken advantage of by using larger filters which could add significant computational load. Based on these findings a patch size of 512×512 pixels was selected. This represented the most efficient size for balancing processing time, and optimizing receptive field size with respect to computational demand based on the information available at the time. However there were some unexpected problems that may have some basis in the patch size and the non-overlapping patch creation process. Upon inference we see that errors are more likely to occur along border regions. Our evaluation metrics do not capture spatial information regarding incorrect pixel-level predictions, but when reviewing the inferred patches reassembled into the context of the whole slide we observe errors associated with patch borders as seen in Figure 5.11.

One possible explanation for this relates to the patch size. The division of the source images into patches is a naive process that does not account for features along the border regions that will be “split” into different patches. This results in many “fractional” cells and other structures. Since most of these border regions occur more frequently in classes other than the epidermis, border region accuracy in epidermal regions seems to suffer. Larger patch sizes would reduce the overall number of border regions and with some modifications to the network may increase the receptive field size to reduce the problem. In addition to increasing the patch size, in future iterations it may be useful to experiment with overlapping patch borders, or using a random cropping approach, such that during training,

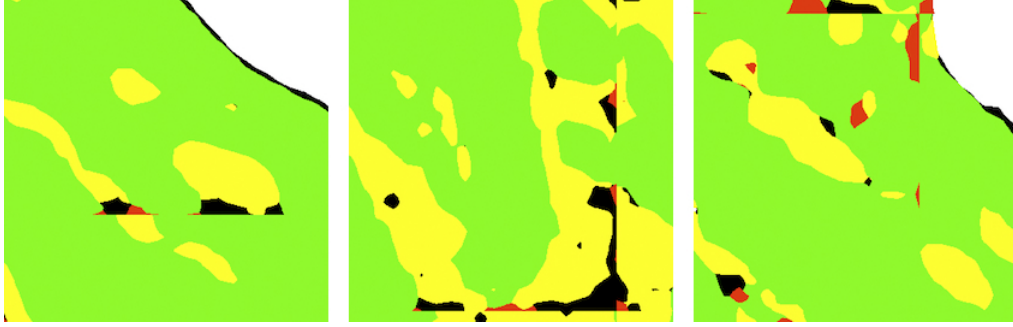


Figure 5.11 – Examples showing errors as patching artifact (black and red pixels are misclassified). Note the black and red pixels align along the border of the patch sent for inference. In the left-most image adjacent patches seemed to accurately infer the dermis (yellow), but this region is interrupted by erroneous inference at the border.

smaller patches are selected from random areas of the larger input patch. Of course this could be done on the inference images, alone but it would add unnecessary time to every inference test, thus it is preferable to solve the problem within the network itself.

Dermis Regions

The dermal layer was challenging to annotate due to the variety of normal structures found within the dermis, and also due to tumour-related alterations of the tissue structure. While the majority of the dermis consists of collagen bundles, the dermis also contains adnexal structures, such as hair follicles, eccrine glands, sebaceous glands, nerve bundles, adipose tissue, blood vessels, and arrector pili muscles. Depending on the anatomic location, these structures can be numerous within the normal dermis and it would be challenging to simply omit them from the dermal layer. The dermis could also demonstrate scar formation and other changes related to previous procedures or biopsies, which again alters the structure of the dermis. By its nature, the destructive growth of the tumour could cause compressive changes and peritumoural stromal alterations. Consequently, to capture the full range of the dermis, it was decided that anything between the epidermis and the subcutaneous adipose tissue would be included in the dermis layer. It should be noted that while the epidermis serves as the superficial boundary of the dermis, the lower (deep) boundary of the dermis is traditionally the adipose tissue. Here arose another challenge, as the dermal collagen merges with the fibrous septae of the subcutaneous adipose tissue. Therefore an arbitrary distinction was made as to the deep boundary of the dermis. More work is needed to devise a strategy for the annotation of this area in the future.

Inference Times

Inference times presented include the time taken to run the forward pass, parse the output blob for the highest scoring class for each pixel, crop the output, and generate the image. This is done for each patch. In real-world usage the inference takes much longer due to the overhead of setting up the network, patch pre-processing, and file IO. This overhead adds up to over 500ms per patch using our unoptimized python code executed while other jobs were executing on the CPU. In a WSI with 2000 patches the total time could be over 15 minutes or potentially over 30 minutes for a very large slide. We expect that the overhead processing could be significantly reduced through optimization and parallelizing aspects of file I/O or using C++ for I/O. Further reduction is expected when using increased patch size.

Metrics While it varies based on the dataset, the quantitative metrics we used to evaluate performance of our experiments is not a clear indicator of the “real world usefulness” of the result or how much better some incremental improvement would be. As seen with the FCN-AlexNet results good performance on most metrics can be achieved while the qualitative result is poor. We have been reluctant to make claims based on these metrics.

5.4 Summary

In this chapter we have included the details of our training process, discussing the major components of the process including hyperparameter selection, transfer learning, and the loss function. We have presented the results of our experiments culminating in a final demonstration of how our network design performs on whole slide images when trained on our dataset. Based on the results, we provided a discussion of the various challenges we observed and where possible, provided some insight as to how they might be remedied in the future.

Conclusions

6.1 Introduction

The research in this thesis arose after searching the deep learning literature in medical imaging to the realization that little work had been done in the area of deep learning and cutaneous melanoma diagnostics in digital pathology. Given the prevalence of this cancer and its potential severity, we wanted to take the opportunity to explore if and how modern machine learning techniques could be applied in this important area with an eye towards future computer aided diagnostic technology in a clinical environment, just as we have seen with other cancers such as breast and prostate.

6.2 Summary

Our hypothesis was that we could build a deep learning model capable of segmenting the three most important tissue types used to diagnose and stage melanoma on whole slide images.

In **Chapter 2**, we reviewed much of the literature focusing on work that has been done in the application of semantic segmentation via convolutional neural networks to medical imaging. We divided the review into two key areas that reflect the work done in this thesis. First we reviewed works that investigate convolutional neural network architectures for segmentation problems in medical imaging and digital pathology. We found a small set of canonical architectures, proven on industry standard datasets, that served as the basis

for adaptation by researchers in medical imaging. From this review we selected the FCN architecture as the basis for the work in this thesis.

Then, we identified similar works in the narrow area within medical imaging of cancer/disease diagnostics and detection. In this part of the review we covered some successful works that perform binary segmentation on whole slide images which were helpful in determining some elements of our workflow. Though this review demonstrated that many details regarding architectural and training methods can be highly dataset specific, the review provided some bounds for key decisions such as patch size, magnification, and various hyperparameters.

Finally we summarized the important competitions in medical imaging where either semantic segmentation is included as a task or the competition included relevant details regarding how their dataset was curated, the type of network, and other factors like the use of CRFs.

In **Chapter 3**, we provided a fast-paced overview of some of the mathematical principles of neural networks and deep learning. We included a discussion of convolutional networks with some details on the layers at the core of most CNN architectures. We provided some resources for more in-depth study of the topics we covered.

In **Chapter 4**, we presented our major contributions starting with a description of the dataset and the details of its curation. We described the intricacies of the annotations and the related technical underpinnings. With the 50 patient cases we chose a 70-15-15 percent split between train-validation-test. We outlined the data augmentation strategy used to manage the class imbalance consisting of the undersampling of background and NDI labelled pixels and the oversampling of the minority class, epidermis.

We also introduced our multi-stride fusion fully convolutional network architecture and the process by which it was arrived at. This architecture was intended to extract as much information as possible from each patch sent to the network without increasing the depth of the network or increasing filter sizes.

Finally in **Chapter 5**, we presented our results. We showed the quantitative approach to validating our network design as we compared progressive designs and final results against the baseline network. We showed our qualitative results including successful and

unsuccessful patch inference examples and the inference results for four of our test slides as whole slide image segmentation masks and detection maps.

6.3 Contributions

In this thesis we have developed an end-to-end, slide-to-prediction FCN based machine learning workflow for automated melanoma cancer diagnostics and understanding. We have curated a new dataset with high quality annotations provided by a practicing, expert pathologist. To our knowledge this is the first work to use deep learning techniques for melanoma diagnosis in digital pathology.

Using this previously unavailable technology we have trained a model to detect and localize three tissue types required for the diagnosis and staging of cutaneous melanoma. With this system we are able to ingest a gigapixel whole slide image and output a full resolution inference image that labels each pixel. The system is capable of segmenting the structures required for measuring two important diagnostic metrics known as Breslow thickness, and the Clark level (please see Appendix B).

This success answers our primary research inquiry which sought to determine if segmenting multiple imbalanced tissue types simultaneously from complex melanoma cases in WSIs was possible and if so, how close to human level accuracy can be achieved. We also demonstrate that the inference time for a whole slide gigapixel image can be completed in a practical timeframe.

In developing this technology we feel we have made the following contributions:

1. We have curated a new dataset containing 50 fully annotated whole slide images of primary cutaneous melanoma or cutaneous metastatic melanoma.
2. We presented a method for the detection of melanoma tumours in whole slide H&E pathology images.
3. We developed the basis of a slide understanding framework to characterize diagnostically important structures for the histomorphometry and staging of melanoma cases through semantic segmentation using fully convolutional neural networks.

4. We have developed a novel multi-stride multi-scale fusion fully convolutional architecture designed to mimic human approaches to processing the rich information present in a microscopic tissue specimen.

6.3.1 Limitations

Despite this encouraging result there are a number of limitations to this work in its current state. Here we describe the major limitations and weaknesses:

Dataset

There are two areas of inconsistency in how the dataset was annotated that has impacted the quality of our results:

- The dermis layer annotation strategy was not clearly defined from the outset of the annotation process. The dermis includes many different structures and itself can be divided into two high-level anatomical layers known as the papillary dermis, and the reticular dermis. This, taken with its varying states of normality depending on the proximity to tumour regions makes it particularly difficult to classify. It was a mistake to group all of these structures into a single class as we found the network struggled to accurately infer the dermis class in challenging areas. During the planning stage this made sense since the Clark level (which requires the dermis to measure) metric does not require discrimination of the varying structures across the dermal area. In reality the dermis is so varied in appearance that more refinement is needed to properly train a model to recognize these areas with greater accuracy. This affects our overall results and as we will discuss in future work will be problematic when measuring the Clark level if this is not addressed.
- The NDI class was also inconsistently annotated due to changes that occurred part way through the annotation process. Due to the semi-automated process by which the NDI annotation was applied, pixels that would more appropriately be labelled as background or in some cases as the surrounding class instead were labelled as NDI (see Figure 6.1). This creates a “hyper-supervised” annotation where the resolution of the labels is at the single-pixel level. Since the network is unable to make single pixel predictions at this granularity these pixels will be wrongly inferred at test-time and on

patches where there are many such instances the metrics are unnecessarily lower than need be. There are enough patches that exhibit this flaw that the maximum accuracy possible is limited. This phenomenon was detected about 70% of the way through the production of annotated images and steps were taken to reduce this effect. This issue will be addressed in future work.

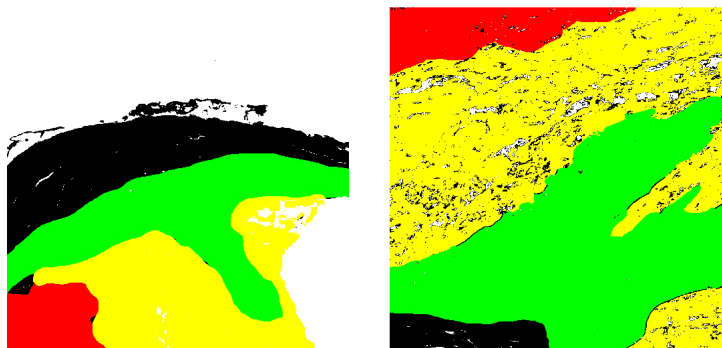


Figure 6.1 – Comparison of aggressive an NDI annotation (right) with the alternate strategy of labelling darker pixels in background areas as the surrounding class or as background instead of NDI in gold standard annotated patches (left). Note the black pixels within the dermis (yellow area) of the image on the right versus the lack of black pixels in the dermis area of the image on the left.

Additional data augmentation: There was still some large variance in network performance depending on how the data was split between training, validation, and test. This could be related to the significant difference in pixel dimensions between slides, when doing randomized sorting into the respective data sets folds. Poorly sorted dataset trials continue to suffer from some overfitting and poor performance. More tweaking of the augmentation strategy and the possible manual selection of cases for the split could be warranted. As a result of these factors the final results were produced by a network trained on approximately 66% of the available data in the training set. Most importantly we need to implement colour intensity modulation to account for the huge variation in stain intensity across the WSIs in our dataset (see Figure C.1). Additionally, we need to do a better job balancing the dataset classes to combat the challenges with the dermis and NDI classes.

Single reviewer: The annotations were created and reviewed by a single pathologist. It would be preferable to have each slide reviewed or ideally annotated by multiple independent pathologists, preferably spanning multiple institutions. This would minimize any interpretation bias introduced by a single annotator, reduce errors, and improve the overall

quality of the dataset.

Network Design

There were some concessions made in the design of our network in order to accommodate the pre-trained weights from the FCN networks we used that were previously trained on PascalVOC. This was done as a trade-off to achieve high quality results with significantly less training time. This limited our network’s design with respect to filter sizes in convolutional layers. By not experimenting with filter sizes and numbers we have not fully explored the potential of our designs.

Adapting filter sizes and training the network from scratch on our dataset using one or more semantic segmentation datasets such as PascalVoC would allow more control of receptive field size and the overall feature learning in filters. We expect such changes to bring improvements to the overall accuracy moving towards better than human accuracy in future iterations.

We should state that improvement of the network goes hand-in-hand with patch-size tuning. Some of our results indicate that a larger patch-size could improve results. Our patch-size is likely too small. The subset of the training dataset that was used to test patch-sizes was too small to fully evaluate performance in this area. As a result, additional experiments are required to find an optimal filter and patch size combination that does not significantly increase compute demand. This could inform how patch size is determined for other medical datasets that use gigapixel images. Taking the result of these two approaches and where possible applying it to existing datasets and competitions would allow our approach to be further validated.

Metrics and Evaluation

While we have used standard metrics for the evaluation of semantic segmentation these metrics do not provide much context to judge their meaning with respect to performance. Since we have only compared performance with two existing networks and no other researchers have had the opportunity to test against this dataset our results lack a high quality benchmark.

Also we have not benchmarked our architecture on other segmentation datasets to quantify its performance on a broader scale. At this time we can only make claims regarding the strength of our architecture in the context of our dataset.

Pathologist’s agreement metrics do not consider cases on the “pixel” level, but rather on the presence and type of cancer with relevant descriptions. Therefore in our task it is difficult to measure segmentation performance compared to humans.

6.4 Future Work

Is it possible for a machine to diagnose cancer in a clinical setting with little or no oversight by a human? Have a machine not only indicate the presence of cancer but also describe in detail the justification for the diagnosis and the nuances of the disease in a formal report? Could a machine learn to detect features in tissue from WSIs that provide currently unknown prognostic information? These are the questions that this work provides the basis for answering as part of longer term research. That said, there are a number near term goals that merit further exploration:

- The near term logical progression of the work we have presented is the addition of two diagnostic metrics (Clark level, Breslow thickness). Using the technology developed herein, algorithms could be developed to generate these tissue measurements based on pixel distance calculations on segmentation masks we can already generate at inference time. This would automate another step in the diagnostic process. Additionally it may be possible to learn these measurements within the network. With this in mind, where applicable, we have included a Breslow layer in the annotation source files of the dataset. This layer includes markup for the position and length of the Breslow measurement.
- Based on the large effect small changes to the data augmentation and annotation approaches had on network performance during our experimentation, we believe there is room for further optimization of the dataset. We have curated the dataset with “source” files. Using these files it is possible to both refine the existing annotations and to provide new annotations for structures not currently included. For example, the previously discussed issues with the dermis class and the other annotation inconsistencies could be resolved. In addition to this, experiments using class border regions as done with the PascalVOC dataset should be investigated.
- We plan to release the raw dataset source files, however we would like to release the

dataset including patches in a standardized format. This would ideally represent the optimal or near optimal combination of the variables such as patch size that contribute to the quality of the result. This will require the execution of the various future experiments discussed throughout this work, as well as defining the logistics of its formal distribution/availability.

- While we do not currently have access to any other information beyond the slide images regarding the patient cases, it may be possible to get more information regarding health outcomes and some patient details such as age and gender from TCGA. Using this information combined with the technology we have developed, it may be possible to build a more robust model capable of providing more prognostic details regarding a given patient's disease, such as treatability and life expectancy. This type of information may be available in the slides on a level where genetic expressions of disease characteristics are visible as morphological features on the slide but are yet to be described in the literature. This would require a large unsupervised component and represents how the work in this thesis and other similar works in the field of diagnostic medicine could be used in a capacity of scientific discovery in addition to improving and assisting in practical tasks.
- **Key Application** While we look forward to a direct augmentation of the human diagnostic process using works similar to that presented herein, the application to quality assurance and control (QA/QC) within a pathology laboratory represents a great opportunity. While the details of QA/QC processes vary widely between institutions, most quality assurance schemes utilize either prospective or retrospective review of past cases for diagnostic discrepancies. This requires both recognition and investment of medical personnel from medical laboratories to provide the resources required for such a program. We foresee a role for this work and those of its kind to act in a pre-screening or post-screening capacity, potentially to flag or screen cases for problematic aspects and prioritize or facilitate for human review. Working with patient cases in a QA/QC context is a logical application area since there is no direct connection to diagnosis and treatment, and therefore no direct impact on patient safety in the case of a failure. While this work is not sufficiently developed to automate such aspects of QA/QC we feel this is a likely direction for this work as a precursor to inclusion in a

clinical diagnostic workflow.

APPENDICES

Appendix A

Pathologist's diagnostic process

Pathologists begin the diagnostic process by examining a slide at low magnification, typically using objective magnifications of $2\times$ or $4\times$, for distortions in tissue architecture. At this magnification, not only does the pathologist locate the areas of interest, but also generates a differential diagnosis and may also note other pertinent features, such as involvement of surgical margins by tumour. More detailed examination of these areas of interest is then performed at a medium ($10\times$) or high ($20\times$ or $40\times$) magnification as needed to confirm additional details about a lesion, including assessment of cytologic features. While the diagnostic process is often straight forward, challenging cases may require multiple iterative examinations at both high and low magnification in order to classify a lesion and provide a diagnosis.

Final interpretation of a slide may require other information not available on the slide itself, such as a review of the patient's clinical history, operative notes, or associated radiologic findings, review of macroscopic findings (i.e., gross description), and review of prior specimens. Ancillary tests, such as additional step sections, special stains, or immunohistochemical studies, may also need to be incorporated into the final diagnostic report [87].

Appendix B

Background on Pathology Metrics

Pathology reports not only contain a diagnosis but also list pertinent features for the prognostication and management of a given disease. In cutaneous melanoma, the most important prognostic factor is the maximum tumour thickness, sometimes referred to as the Breslow thickness. According to the College of American Pathologists melanoma reporting protocol, considered throughout North America as the standard for reporting [88], this is typically measured at a right angle to the adjacent normal skin. Where the epidermis is intact, the superficial point of reference is the upper edge of the granular layer of the epidermis; when the epidermis is lost, i.e., ulcerated, the base of the ulcer is used as the upper point of reference. The deepest point of the main mass of the tumour is taken as the deep point of reference. In the upcoming 8th revision of the main staging system, provided by the American Joint Committee on Cancer [89], this measurement should be accurate to the nearest 0.1mm. The importance of the Breslow thickness cannot be overemphasized, as it forms the basis of the tumour ('T') staging as follows:

pT1: Melanoma 1.0 mm or less in thickness, ulceration status unknown or unspecified

pT1a: Melanoma <0.8 mm in thickness, no ulceration

pT1b: Melanoma <0.8 mm in thickness with ulceration OR melanoma 0.8 to 1.0 mm in thickness with or without ulceration

pT2: Melanoma >1.0 to 2.0 mm in thickness, ulceration status unknown or unspecified

pT2a: Melanoma >1.0 to 2.0 mm in thickness, no ulceration

pT2b: Melanoma >1.0 to 2.0 mm in thickness, with ulceration

pT3: Melanoma >2.0 to 4.0 mm in thickness, ulceration status unknown or unspecified

pT3a: Melanoma >2.0 to 4.0 mm in thickness, no ulceration

pT3b: Melanoma >2.0 to 4.0 mm in thickness, with ulceration

pT4: Melanoma >4.0 mm in thickness, ulceration status unknown or unspecified

pT4a: Melanoma >4.0 mm in thickness, no ulceration

pT4b: Melanoma >4.0 mm in thickness, with ulceration

Although the Breslow thickness has superseded the anatomic (Clark) levels in prognostic significance, the Clark level remains as an important predictor of outcome and is typically recorded on melanoma pathology reports. The Clark levels are as follows:

- I Intraepidermal tumour only (i.e., melanoma in situ)
- II Tumour present in but does not fill and expand papillary dermis
- III Tumour fills and expands papillary dermis
- IV Tumour invades into reticular dermis
- V Tumour invades subcutis

Appendix C

Tissue Variation

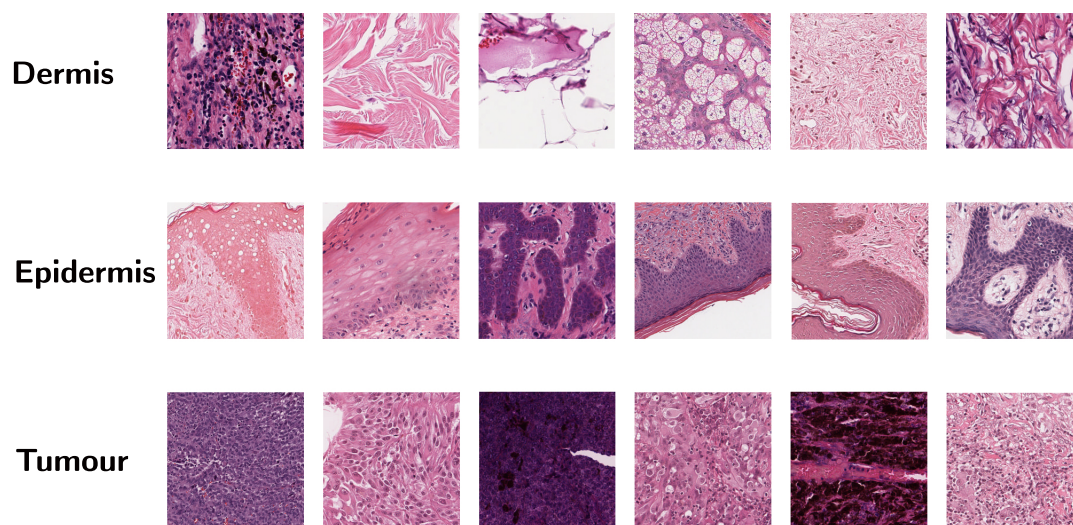


Figure C.1 – A selection of patches to illustrate the variation between structures and staining for each tissue type in our data set.

Appendix D

Anatomy of the Skin

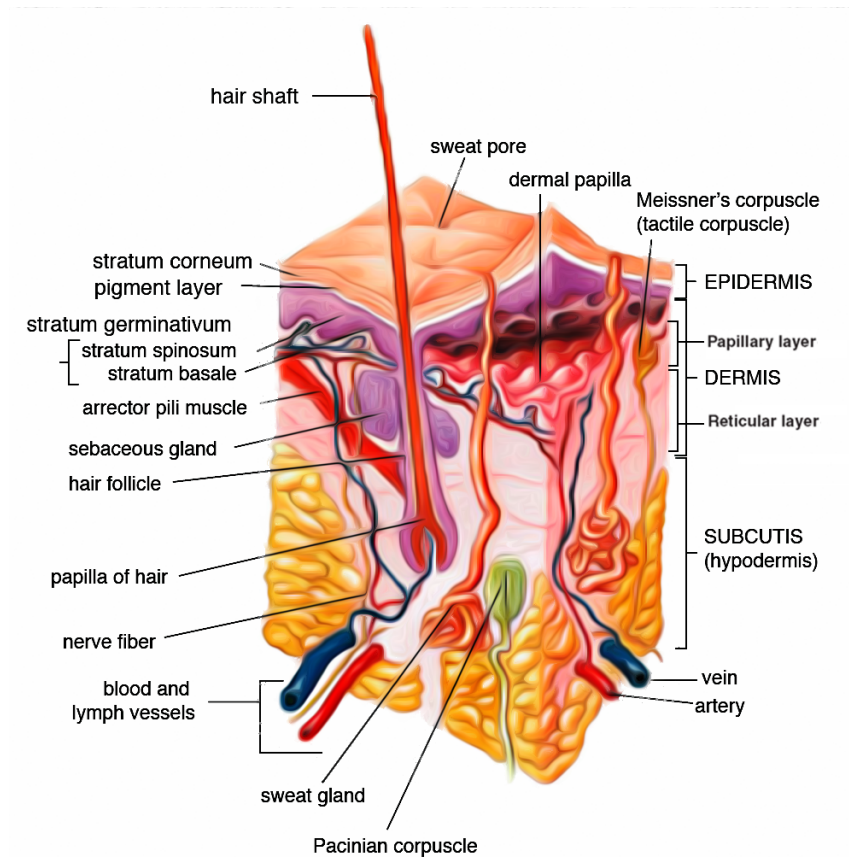


Figure D.1 – High level view of the major structures of skin anatomy. Note that our annotations of the epidermis do not include the stratum corneum. Diagram adapted from public domain image based on L.M. Anthony [90].

Appendix E

Baseline for Scale

For completeness we include the results of a single stride, two stream network (Multi-Scale-1). In this network, one stream computes FCN-16s using full resolution patches and the second stream computes FCN-16s using patches downsampled by a factor of two as in our multi-stride, multi-scale network via max-pooling. We include these results to verify that the addition of multi-scale information alone does not perform better than our proposed networks that include multi-stride information.

We found that the Multi-Scale-1 (MS1) network was trained with a lower validation accuracy than the baseline FCN-16s (85.695 versus 85.720) using an identical training environment. Using our standard evaluation method we found that MS1 scored well compared to FCN-16s and came closer in score to our networks (see Table E.1). The qualitative evaluation was nearly identical to FCN-16s in our tests. We suspect that the higher scores achieved by MS1 in some cases is the result of inferring the majority of pixels in an area, but not accounting for finer details within (see Figure E.1). This is similar to FCN-AlexNet results, but to a lesser degree.

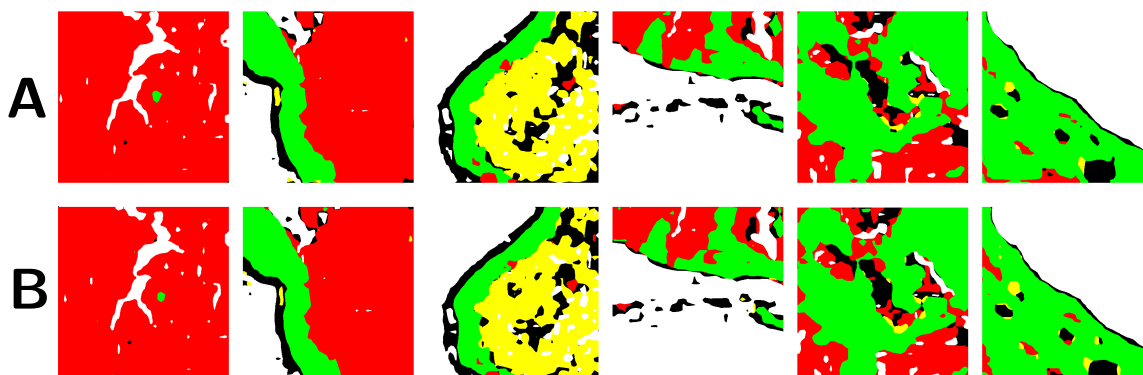


Figure E.1 – Qualitative comparison of a baseline multi-scale network with FCN-16s using a selection of inference patches. Row A shows results from FCN-16s, row B shows results from MS1.

Table E.1 – Evaluation results using the Multi-Scale-1 network

Multi-Scale - 1

Case ID	PA	mPA	mIoU	fwIoU	Score	Time
BF-A5EO	0.9566	0.6724	0.6286	0.9393	0.6505	61.7193
D9-A3Z4	0.9047	0.5861	0.5397	0.8558	0.5629	133.6817
EB-A3XE	0.9123	0.5984	0.5541	0.8708	0.5762	125.5430
EE-A2GO	0.8984	0.6004	0.5556	0.8679	0.5780	164.6892
EE-A182	0.8977	0.6249	0.5589	0.8835	0.5919	205.5935
ER-A19H	0.8792	0.6993	0.6408	0.8430	0.6700	262.5221
WE-A8K4	0.8401	0.5473	0.5231	0.8295	0.5352	222.5077
Mean:	0.8984	0.6184	0.5715	0.8700	0.5950	168.0367

References

- [1] V. Kumar, A. Abbas, N. Fausto, and J. Aster, *Robbins and Cotran: Pathologic Basis of Disease*. Elsevier Saunders, 2014.
- [2] M. D. Abeloff, J. O. Armitage, J. E. Niederhuber, M. B. Kastan, and W. G. McKenna, “Ch 73. Melanoma,” in *Abeloff’s Clinical Oncology*. Philadelphia, PA: Churchill Livingstone Elsevier, 2008, pp. 1229–1252.
- [3] American Cancer Society, “Cancer facts & figures 2016,” [Online]. Available: <https://www.google.ca/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwiRpoKf0rPTAhUn6oMKHQEBDt8QFggiMAA&url=https%3A%2F%2Fwww.cancer.org%2Fresearch%2Fcancer-facts-statistics%2Fall-cancer-facts-figures%2Fcancer-facts-figures-2016.html&usg=AFQjCNGv0sTs4bCkwb61lGt0Cwv9U3BKLw>. Accessed: 04/04/2017
- [4] [Online]. Available: <https://www.aimatmelanoma.org/about-melanoma/melanoma-stats-facts-and-figures/>.” Accessed:
- [5] A. Glazer, R. Winkelmann, and A. Farberg. . . , “Analysis of trends in US melanoma incidence and mortality,” *Journal of the American Medical Association dermatology*, vol. 153, no. 2, pp. 225–225, 2017.
- [6] O. Orzan, A. Sandru, and C. Jecan, “Controversies in the diagnosis and treatment of early cutaneous melanoma,” *Journal of medicine and life*, vol. 8, no. 2, p. 132, 2015.
- [7] A. Goodson and D. Grossman, “Strategies for early melanoma detection: approaches to the patient with nevi,” *Journal of the American Academy of Dermatology*, vol. 60, no. 5, pp. 719–719, 2009.
- [8] A. Madabhushi and G. Lee, “Image analysis and machine learning in digital pathology: Challenges and opportunities,” *Medical Image Analysis*, vol. 33, pp. 170 – 175, 2016.
- [9] J. Schmidhuber, “Deep learning in neural networks: An overview,” *Neural Networks*, vol. 61, pp. 85–117, 2015.

- [10] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [11] A. Krizhevsky, I. Sutskever, and G. Hinton, *ImageNet classification with deep convolutional neural networks*. Curran Associates Inc., 2012.
- [12] R. Olga, D. Jia, S. Hao, K. Jonathan, S. Sanjeev, M. Sean, H. Zhiheng, K. Andrej, K. Aditya, B. Michael, C. B. Alexander, and F.-F. Li, “ImageNet large scale visual recognition challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
- [13] C. Ciresan, Dan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” *CoRR*, vol. abs/1202.2745, 2012.
- [14] A. Cruz-Roa, A. Basavanahally, F. Gonzalez, H. Gilmore, M. Feldman, S. Ganesan, N. Shih, J. Tomaszewski, and A. Madabhushi, “Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks,” *SPIE Medical Imaging*, vol. 9041, pp. 904 103–904 103, 2014.
- [15] M. Ertosun and D. Rubin, “Automated grading of gliomas using deep learning in digital pathology images: A modular approach with ensemble of convolutional neural networks,” *AMIA Annu Symp Proc*, vol. 2015, pp. 1899–1908, 2015.
- [16] X. Jun, L. Xiaofei, W. Guanhao, G. Hannah, and M. Anant, “A deep convolutional neural network for segmenting and classifying epithelial and stromal regions in histopathological images,” *Neurocomputing*, vol. 191, pp. 214 – 223, 2016.
- [17] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014.
- [18] D. C. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, “Deep neural networks segment neuronal membranes in electron microscopy images proceedings of the 25th international conference on neural information processing systems,” *NIPS 12*, pp. 2843–2851, 2012.
- [19] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” *CoRR*, vol. abs/1505.04597, 2015.

- [20] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3431, 2015.
- [21] D. Z. Matthew, K. Dilip, W. T. Graham, and F. Rob, *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010.
- [22] Y. Song, L. Zhang, S. Chen, D. Ni, B. Lei, and T. Wang, “Accurate segmentation of cervical cytoplasm and nuclei based on multiscale convolutional network and graph partitioning,” *IEEE Transactions on Biomedical Engineering*, vol. 62, no. 10, pp. 2421–2433, 2015.
- [23] W. Yang, Y. Chen, Y. Liu, L. Zhong, G. Qin, Z. Lu, Q. Feng, and W. Chen, “Cascade of multi-scale convolutional neural networks for bone suppression of chest radiographs in gradient domain,” *Medical Image Analysis*, vol. 35, pp. 421–433, 2017.
- [24] J. Wang, J. MacKenzie, and R. Ramachandran. . . , “A deep learning approach for semantic segmentation in histology tissue images,” *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2016.
- [25] G. Litjens, C. Sánchez, N. Timofeeva, M. Hermsen, I. Nagtegaal, I. Kovacs, C. Hulsbergen-van de Kaa, P. Bult, B. van Ginneken, and J. van der Laak, “Deep learning as a tool for increased accuracy and efficiency of histopathological diagnosis,” *Scientific Reports*, vol. 6, p. 26286, 2016.
- [26] S. Christian, W. Liu, Y. Jia, P. Sermanet, S. E. R. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” *CoRR*, vol. abs/1409.4842, 2014.
- [27] Y. Liu, K. Gadepalli, M. Norouzi, and G. Dahl, “Detecting cancer metastases on gigapixel pathology images,” *CoRR*, vol. abs/1703.02442, 2017.
- [28] V. Gulshan, L. Peng, M. Coram, M. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, R. Kim, R. Raman, P. Nelson, J. Mega, and D. Webster, “Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs,” *Journal of the American Medical Association*, vol. 316, no. 22, pp. 2402–2410, 2016.

- [29] A. Esteva, B. Kuprel, R. Novoa, J. Ko, S. Swetter, H. Blau, and S. Thrun, “Dermatologist-level classification of skin cancer with deep neural networks,” *Nature*, vol. 542, no. 7639, pp. 115–118, 2017.
- [30] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *CoRR*, vol. abs/1502.03167, 2015.
- [31] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–770, 2016.
- [32] M. Drozdal, E. Vorontsov, and G. Chartrand, “The importance of skip connections in biomedical image segmentation,” *International Workshop on Large-Scale Annotation of Biomedical Data and Expert Label Synthesis*, pp. 179–179, 2016.
- [33] M. Drozdal, G. Chartrand, E. Vorontsov, L. D. Jorio, A. Tang, A. Romero, Y. Bengio, C. Pal, and S. Kadoury, “Learning normalized inputs for iterative estimation in medical image segmentation,” *arXiv e-prints*, vol. 1702.05174, 2017.
- [34] H. Rezaeilouyeh, A. Mollahosseini, and M. H. Mahoor, “Microscopic medical image classification framework via deep learning and shearlet transform,” *Journal of Medical Imaging*, vol. 3, no. 4, pp. 044 501–044 501, 2016.
- [35] E. Kim, M. Corte-Real, and Z. Baloch, *A deep semantic mobile application for thyroid cytopathology*, 2016, vol. 9789.
- [36] H. Chang, J. Han, C. Zhong, A. Snijders, and J. Mao, “Unsupervised transfer learning via multi-scale convolutional sparse coding for biomedical applications,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, no. 99, pp. 1–1, 2017.
- [37] J. Andrew, D. Scott, G. Hannah, and M. Anant, “A resolution adaptive deep hierarchical (radhical) learning scheme applied to nuclear segmentation of digital pathology images,” *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, vol. 0, pp. 1–7, 0.

- [38] N. K. M., E. A. R. S., S. K., A. M., and R. N., “Handcrafted features with convolutional neural networks for detection of tumor cells in histology images,” *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, pp. 1029–1032, 2016.
- [39] R. Turkki, N. Linder, P. Kovanen, T. Pellinen, and J. Lundin, “Antibody-supervised deep learning for quantification of tumor-infiltrating immune cells in hematoxylin and eosin stained breast cancer samples,” *Journal of Pathology Informatics*, no. 1, p. 38, 2016.
- [40] V. Mitko, P. J. van Diest, and J. P. W. Pluim, “Cutting out the middleman: measuring nuclear area in histopathology slides without segmentation,” *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016: 19th International Conference, Athens, Greece, October 17-21, 2016, Proceedings, Part II*, vol. abs/1606.06127, pp. 632–632, 2016.
- [41] H. Xu and M. Mandal, “Efficient segmentation of skin epidermis in whole slide histopathological images,” *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 3855–3858, 2015.
- [42] D. Wang, A. Khosla, R. Gargeya, H. Irshad, , and A. H. Beck, “Deep learning for identifying metastatic breast cancer,” *arXiv preprint arXiv:1606.05718*, 2016.
- [43] P. Kyunghyun, S. Hwang, S. Park, M. Kim, and S. Kim, “A unified framework for tumor proliferation score prediction in breast histopathology,” *CoRR*, vol. abs/1612.07180, 2016.
- [44] K. Sirinukunwattana, J. Pluim, H. Chen, X. Qi, P. Heng, Y. Guo, L. Wang, B. Matuszewski, E. Bruni, U. Sanchez, A. Böhm, O. Ronneberger, B. Cheikh, D. Racoceanu, P. Kainz, M. Pfeiffer, M. Urschler, D. Snead, and N. Rajpoot, “Gland segmentation in colon histology images: The GLaS challenge contest,” *Medical Image Analysis*, vol. 35, pp. 489–502, 2017.
- [45] S. Xie and Z. Tu, “Holistically-nested edge detection,” *CoRR*, vol. abs/1504.06375, 2015.
- [46] X. Yan, Y. Li, M. Liu, Y. Wang, M. Lai, and E. I.-C. Chang, “Gland instance segmentation by deep multichannel side supervision,” *CoRR*, vol. abs/1607.03222, 2016.

- [47] B. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, L. Lanczi, E. Gerstner, M. Weber, T. Arbel, B. Avants, N. Ayache, P. Buendia, D. Collins, N. Cordier, J. Corso, A. Criminisi, T. Das, H. Delingette, C. Demiralp, C. Durst, M. Dojat, S. Doyle, J. Festa, F. Forbes, E. Geremia, B. Glocker, P. Golland, X. Guo, A. Hamamci, K. Iftexharuddin, R. Jena, N. John, E. Konukoglu, D. Lashkari, J. Mariz, R. Meier, S. Pereira, D. Precup, S. Price, T. Raviv, S. Reza, M. Ryan, D. Sarikaya, L. Schwartz, H. Shin, J. Shotton, C. Silva, N. Sousa, N. Subbanna, G. Szekely, T. Taylor, O. Thomas, N. Tustison, G. Unal, F. Vasseur, M. Wintermark, D. Ye, L. Zhao, B. Zhao, D. Zikic, M. Prastawa, M. Reyes, and K. Van Leemput, “The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS),” *IEEE Transactions on Medical Imaging*, vol. 34, no. 10, pp. 1993–2024, 2015.
- [48] K. Kamnitsas, C. Ledig, V. Newcombe, J. Simpson, A. Kane, D. Menon, D. Rueckert, and B. Glocker, “Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation,” *Medical Image Analysis*, vol. 36, pp. 61–78, 2017.
- [49] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (VOC) challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.
- [50] L. Liang, G. Wang, R. Zhang, R. Zhang, X. Liang, and W. Zuo, “Deep structured scene parsing by learning with image descriptions,” *CoRR*, vol. abs/1604.02271, 2016.
- [51] A. Janowczyk and A. Madabhushi, “Deep learning for digital pathology image analysis: A comprehensive tutorial with selected use cases.” *Journal of Pathology Informatics*, vol. 7, p. 29, 2016.
- [52] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.
- [53] G. Ian, B. Yoshua, and C. Aaron, *Deep Learning*. MIT Press, 2016.
- [54] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data mining, Inference, and Prediction*. New York: Springer, 2009.

- [55] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” *Proceedings of COMPSTAT 2010*, pp. 177–177, 2010.
- [56] A. Karpathy, “Stanford university cs231n: Convolutional neural networks for visual recognition. course and handouts.” [Online]. Available: <http://cs231n.stanford.edu/syllabus.html>. Accessed: 07/08/2017
- [57] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of Control, Signals, and Systems (MCSS)*, vol. 2, no. 4, pp. 303–314, 1989.
- [58] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.
- [59] L. Prechelt, “Early stopping-but when?” *Neural Networks: Tricks of the trade*, pp. 553–553, 1998.
- [60] S. Park and N. Kwak, “Analysis on the dropout effect in convolutional neural networks,” *Asian Conference on Computer Vision*, pp. 189–204, 2016.
- [61] M. Brubaker, J. Fleet, and A. Hertzmann, “Gradient descent,” [Online]. Available: <http://www.cs.toronto.edu/~mbrubake/teaching/C11/Handouts/GradientDescent.pdf>. Accessed: 09/05/2017
- [62] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” *International conference on machine learning*, vol. 28, pp. III–1139–III–1147, 2013.
- [63] R. Rojas, *Neural Networks: A Systematic Introduction*. New York, NY, USA: Springer-Verlag New York, Inc., 1996.
- [64] R. Socher, “cs224d deep nlp, lecture 5 neural networks and backprop,” [Online]. Available: <http://cs224d.stanford.edu/lectures/CS224d-Lecture5.pdf>. Accessed: 07/08/2017
- [65] M. Lin, Q. Chen, and S. Yan, “Network in network,” *arXiv preprint arXiv:1312.4400*, 2013.

- [66] A. Garcia-Garcia, S. Orts-Escolano, and S. Oprea, “A review on deep learning techniques applied to semantic segmentation,” *arXiv preprint arXiv:1704.06857*, 2017.
- [67] S. Mills, *Histology for Pathologists*. Wolters Kluwer Health, 2012.
- [68] P. Christ, F. Ettlinger, F. Grün, and M. Elshaera. . . , “Automatic liver and tumor segmentation of ct and mri volumes using cascaded fully convolutional neural networks,” *arXiv preprint arXiv:1702.05970*, 2017.
- [69] B. Krawczyk, “Learning from imbalanced data: open challenges and future directions,” *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, 2016.
- [70] S. Yen and Y. Lee, “Cluster-based under-sampling approaches for imbalanced data distributions,” *Expert Systems with Applications*, vol. 36, no. 3, pp. 5718–5718, 2009.
- [71] M. Rahman and D. Davis, “Cluster based under-sampling for unbalanced cardiovascular data,” *Proceedings of the World Congress on Engineering*, vol. III, 2013.
- [72] Z. Zhou and X. Liu, “Training cost-sensitive neural networks with methods addressing the class imbalance problem,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63–63, 2006.
- [73] N. Chawla, K. Bowyer, and L. Hall, “Smote: synthetic minority over-sampling technique,” *Journal of Artificial Intelligence Research*, vol. Volume 16, Issue 1, pp. 321–357, 2002.
- [74] D. Ciresan, A. Giusti, and L. Gambardella, “Mitosis detection in breast cancer histology images with deep neural networks,” *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013*, 2013.
- [75] I. Biederman, R. Mezzanotte, and J. Rabinowitz, “Scene perception: Detecting and judging objects undergoing relational violations,” *Cognitive psychology*, 1982.
- [76] M. Mostajabi, P. Yadollahpour, and S. Gregory, “Feedforward semantic segmentation with zoom-out features,” *CoRR*, vol. abs/1412.0774, 2014.
- [77] W. Liu, A. Rabinovich, and A. Berg, “Parsenet: Looking wider to see better,” *arXiv preprint arXiv:1506.04579*, 2015.

- [78] P. Wang, P. Chen, Y. Yuan, D. Liu, and Z. Huang, “Understanding convolution for semantic segmentation,” *CoRR*, vol. abs/1702.08502, 2017.
- [79] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014.
- [80] M. Thoma, “A survey of semantic segmentation,” *arXiv preprint arXiv:1602.06541*, 2016.
- [81] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, “Scene parsing through ade20k dataset,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- [82] Y. LeCun, L. Bottou, G. Orr, and K. Müller, “Efficient backprop,” (eds) *Neural networks: Tricks of the Trade. Lecture Notes in Computer Science*, vol. 1524, pp. 9–50, 1998.
- [83] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” *arXiv preprint arXiv:1408.5093*, 2014.
- [84] M. Goyal, N. Reeves, S. Rajbhandari, and J. Spragg. . . , “Fully convolutional networks for diabetic foot ulcer segmentation,” *arXiv preprint arXiv:1708.01928*, 2017.
- [85] Y. Bengio, “Practical recommendations for gradient-based training of deep architectures,” *Neural networks: Tricks of the trade*, pp. 437—478, 2012.
- [86] Z. Jia, X. Huang, E. Chang, and Y. Xu, “Constrained deep weak supervision for histopathology image segmentation,” *arXiv preprint arXiv:1701.00794*, 2017.
- [87] A. Phillips, *Pathologist’s diagnostic process: Interview with Dr. I Teo*, 2017.
- [88] D. Frishberg, C. Balch, B. Balzer, A. Crowson, M. Didolkar, J. McNiff, R. Perry, V. Prieto, P. Rao, M. Smith, B. Smoller, and M. Wick, “Protocol for the examination of specimens from patients with melanoma of the skin,” *Archives of Pathology and Laboratory Medicine*, vol. 133, no. 10, pp. 1560–1567, 2009.

- [89] M. B. Amin, F. L. Greene, S. B. Edge, C. C. Compton, J. E. Gershenwald, R. K. Brookland, L. Meyer, D. M. Gress, D. R. Byrd, and D. P. Winchester, *The Eighth Edition AJCC Cancer Staging Manual: Continuing to build a bridge from a population-based to a more “personalized” approach to cancer staging*, 2017, vol. 67(2).
- [90] L. M. Anthony, *Chapter 18. Skin Junqueira’s Basic Histology, 13e*. New York, NY: The McGraw-Hill Companies, 2013.
- [91] B. Kayalibay, G. Jensen, and P. v. d. Smagt, “CNN-based segmentation of medical imaging data,” *CoRR*, vol. abs/1701.03056, 2017.
- [92] Z. Wu, C. Shen, and A. Hengel, “High-performance semantic segmentation using very deep fully convolutional networks,” *arXiv preprint arXiv:1604.04339*, 2016.
- [93] K. Philipp, M. Pfeiffer, and M. Urschler, “Semantic segmentation of colon glands with deep convolutional neural networks and total variation segmentation,” *CoRR*, vol. abs/1511.06919, 2015.
- [94] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *ImageNet: A Large-Scale Hierarchical Image Database CVPR09*. IEEE, 2009.
- [95] Z. Zheng, Y. Cai, and Y. Li, “Oversampling method for imbalanced classification,” *Computing and Informatics*, vol. 34, no. 5, pp. 1017–1017, 2016.
- [96] B. Krawczyk, L. Jelen, A. Krzyzak, and T. Fevens, “Oversampling methods for classification of imbalanced breast cancer malignancy data,” *Computer Vision and Graphics*, pp. 483–483, 2012.
- [97] J. Shotton, J. Winn, C. Rother, and A. Criminisi, “Textronboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context,” *International Journal of Computer Vision*, vol. 81, no. 1, pp. 2–2, 2009.