UNIVERSITY OF SOUTHAMPTON

Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science

Fully Convolutional Neural Network based Histopathology Image Segmentation

by

Yilong Yang

ORCiD: 0000-0002-2595-7883

A thesis for the degree of
Doctor of Philosophy

January 2024

University of Southampton

Abstract

Faculty of Engineering and Physical Sciences
School of Electronics and Computer Science

Doctor of Philosophy

by Yilong Yang

The examination of histopathology images is generally recognised as the *gold standard* for the diagnosis of diseases. Clinical diagnostic practice requires pathologists to follow a descriptive set of guidelines and therefore is prone to suffer from inter-observer variability due to differences in the interpretation of histological patterns. Furthermore, the extremely large size of histopathology images makes it impossible for pathologists to thoroughly inspect every detail of a whole slide image (WSI). Such that the risk of misdiagnosis arises. In this thesis, we aim to develop automatic tools that can objectively analyse and quantify the vast amount of pixel information contained in histopathology images. More specifically, we focus on the automatic tissue type/component segmentation of histopathology images.

To fulfil our objective, we first revisit the UNet family architectures that are widely used in medical imaging and then suggest a novel ensemble framework to tackle the shortcomings existing in UNet++-like architectures. We present a novel stage-wise additive training algorithm that, using ideas from boosting, incorporates resource-efficient deep supervision in shallower layers and takes performance-weighted combinations of the sub-UNets to create the segmentation model. To ensure the effectiveness of the ensemble, we designed a scheme in which the diversity of features is guaranteed.

On the other hand, we identify the loss of magnification information as a key barrier to the application of automatic computational pathology tools. In this regard, we explore scale equivariant methods that possess the capability of achieving consistent segmentation to bypass the loss of magnification information. We present the Scale-Equivariant UNet (SEUNet) for image segmentation by building on scale-space theory. The SEUNet contains groups of filters that are linear combinations of Gaussian basis filters, whose scale parameters are trainable but constrained to span disjoint scales through the layers of the network. By encoding scale equivariance into CNNs, histopathology images presented at different scales are more likely to be consistently segmented.

Furthermore, due to the inherent rotation symmetry of histopathology images, it is desirable for CNNs to be rotation-equivariant. This guarantees that features transform as expected with the rotation of the input; thus a consistent segmentation can be produced

regardless of the presenting angle of the image. To leverage this prior knowledge, we extend our proposed scale equivariant UNet to a joint rotation-scale equivariant model. By introducing weight-sharing between multi-scale and multi-orientation filters, the joint equivariance of rotation and scale is achieved, yet the number of trainable parameters is dramatically decreased when compared with conventional CNN filters. The proposed rotation-scale equivariant method shows the state-of-the-art generalisation performance in scenarios wherein scale and orientation variations of images exist.

Contents

List of Figures

List of Tables

Declaration of Authorship

I declare that this thesis and the work presented in it is my own and has been generated by me as the result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a research degree at this University;
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
3. Where I have consulted the published work of others, this is always clearly attributed;
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
5. I have acknowledged all main sources of help;
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
7. None of this work has been published before submission

Signed:..... Date:.....

Acknowledgements

I would like to express my sincere gratitude to all the people who supported me during my PhD study. First and foremost, I would like to thank my supervisors, Dr. Sasan Mahmoodi and Dr. Srinandan Dasmahapatra, for their constant encouragement and guidance. They have walked me through all the stages of my PhD study. Thanks to their patience in helping and teaching me to acquire invaluable knowledge from which I will benefit in a lifetime.
Secondly, I am very grateful to my friends and colleagues at the Vision, Learning and Control (VLC) group for their companionship, valuable discussions and suggestions over the memorable four years.

Finally, with my love and gratitude, I would like to dedicate this thesis to my family who always give me unreserved support and selfless love. It is my best luck to have them in my life. A special thanks must go to my girlfriend Jingjing who has shown her continual love and support over the last few years. Her ability to make me laugh and to pick me up when times are tough has pushed me to keep going and remain positive during the course of my PhD.

I extend my gratitude to the China Scholarship Council, for its financial support during this academic journey.

To my grandparents.

Definitions and Abbreviations

AI	Artificial Intelligence
CAD	Computer-Aided Diagnosis
CL	Cascade Learning
CNN	Convolutional Neural Network
CKA	Centered Kernel Alignment
CV	Cross Validation
DL	Deep Learning
DNN	Deep Neural network
FCN	Fully Convolutional Networks
GPU	Graphic Processing Unit
GCNNs	Group Convolutional Neural Networks
H&E	Hematoxylin and Eosin
IoU	Intersection over Union
mIoU	mean Intersection over Union
ReLU	Rectified Linear Unit
WSI	Whole Slide Images

Chapter 1

Introduction

According to the statistics reported by the World Health Organization (WHO) in 2020, cancer is a leading cause of death worldwide, accounting for nearly one in six deaths. Moreover, it is predicted there will be 28 million new cancer cases worldwide each year by 2040, if incidence remains stable and population growth and ageing continue in line with recent trends [83]. Early diagnosis of cancer enables patients to receive timely treatment and then significantly decreases the probability of death [42]. Although there are many methods that can be used to diagnose cancer, the examination of histopathology images is still regarded as the gold standard [90]. Cancer tissue identification on histopathology images is currently performed by pathologists. However, this procedure is labour-intensive, time-consuming, error-prone and relies extensively on pathologists' professional experience. In 2018, a survey of the UK histopathology workforce revealed a serious shortage of pathologists. In detail, only 3% of the departments have enough pathologists to meet clinical demands, 45% of departments have to send work away to cope with demand and half of the departments have to use locums [78]. Moreover, Cancer Research UK's report indicates that without targeted action and investment, the number of histopathologists is forecast to reduce from the existing shortfall by an additional 2% by 2029 [36]. To narrow the gap between increasing diagnosis demands and decreasing pathology workforce, the community resorts to AI-Assisted pathological diagnosis. A recent survey study forecasts that AI would be routinely and impactfully used within anatomic pathology laboratories and pathologist clinical workflows by 2030 [12]. In this research, we focus on developing computer-aided algorithms for assisting quantitative assessment of histopathology images. In this chapter, the background and research objective of this research are introduced and then the outline of the thesis is provided.

1.1 Background

1.1.1 Histopathology

Histopathology refers to the microscopic examination of tissue in order to study the manifestations of disease. In clinical medicine, histopathology refers to the examination of a biopsy or surgical specimen by a pathologist, after the specimen has been processed and histological sections have been placed onto glass slides. Before this examination can take place, the tissue must be appropriately prepared. This preparation consists of the following steps:

- **Fixation.** The fixation process preserves the structure of the tissue sample and prevents it from degradation by introducing chemicals that stimulate cross-linkage between proteins. The most usually used fixative is Formalin.
- **Dehydration.** involves immersing the specimen in increasing concentrations of ethanol to remove the water and formalin from the tissue. Afterwards, xylene is used to remove the ethanol.
- **Embedding.** the tissue sample in a solid material for the purpose of thin sectioning in the next stage. Paraffin wax or plastic resin is often used at this stage.
- **Sectioning.** Placing the specimen on a microtome and cutting it into sections of about 5 m, which is the optimal thickness for staining. Then mounting the sections on glass slides.
- **Staining.** Most cells are transparent and appear almost colourless when unstained. Staining increases contrasts of different components, making tissue structures more visible and thus easier for visual examination. Haematoxylin and Eosin (H&E) are the most commonly used dyes that stain nuclei dark blue/purple and stain extracellular matrix and cytoplasm pink.

During the preparation process, there can be a large variation in the appearance between different stained tissue samples. For example, differences in a) thickness of sections, b) temperature, c) stain concentration and d) duration of staining, can lead to stain colour variation. The stain appearance variation may not impact pathologists' decisions when diagnosing tissues; however, algorithms are usually negatively affected by it [102].

Afterwards, visual examinations of tissue slides can be done by pathologists under microscopes. This examination procedure involves moving glass slides, adjusting illumination, tuning magnification, etc. Finally, a medical diagnosis is formulated as a pathology report describing the histological findings and the opinion of the pathologist.

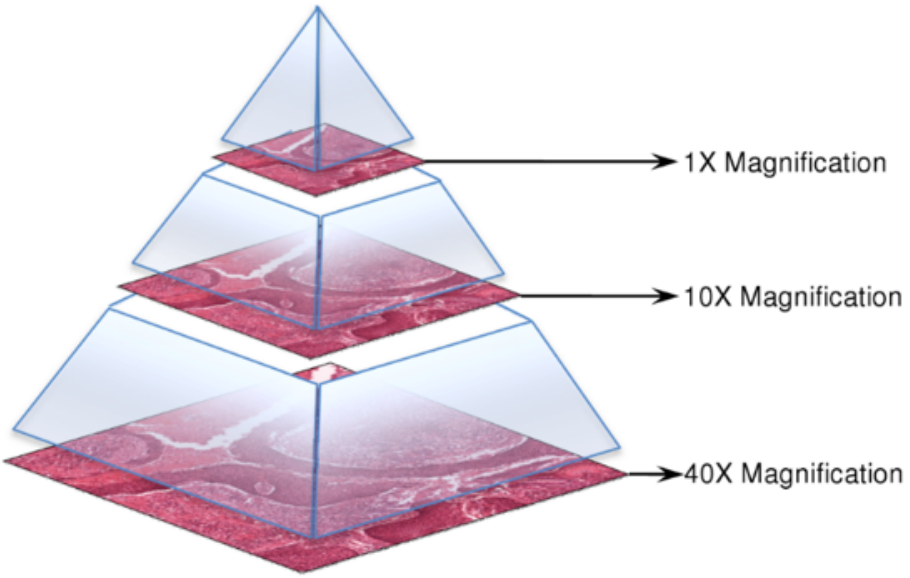


FIGURE 1.1: An illustration of the pyramidal structure a whole slide image containing different magnifications. Image source:

1.1.2 Digital Pathology

Digital pathology includes the acquisition, management, sharing and interpretation of pathology information in a digital environment. Through the use of microscopy slide scanners, glass slides are converted into digital slides, known as Whole Slide Images (WSIs), that can be viewed, managed, shared and analysed on digital platforms. WSIs are usually stored in a pyramid format with each level of the pyramid representing a unique magnification level (see Figure 1.1 for example). Typically, a WSI at 40 objective magnification contains 20 billion pixels and requires approximately 56GB of memory [22]. Due to the huge size of the WSI, it is already challenging to load the entire WSI into GPU or GPU memory with standard image libraries, let alone analyze the entire image all at once. Fortunately, there are some libraries such as OpenSlide [38] that provide high-level interfaces for efficient data reading and retrieving from WSIs. There are also some platforms like Orbit [99] and QuPath [5] given users on-slide visualisation functions.

1.1.3 Computational Pathology

The advance of digital pathology technology brings researchers the opportunity to explore computational algorithms to analyse patterns of WSIs, which can help to build computer-aided diagnosis (CAD) systems and ultimately replace pathologists. Computational Pathology (CPath) is a branch of pathology where multiple sources of patient information including pathology image data and meta-data are combined to extract

patterns and analyse features for the study of disease [1]. As concluded in a recent survey, although very few algorithms are currently in routine clinical use in pathology, experts agreed that artificial intelligence would be routinely and impactfully used within anatomic pathology laboratories and pathologist clinical workflows by 2030 [12]. In general, research directions of CPath include (1) pre-processing, (2) segmentation, and (3) Diagnosis/Prognosis [116]. Below we provide a brief introduction to each sub-field.

Pre-processing. As mentioned in Section 1.1.1, colour variations exist in glass slides due to variations in the tissue preparation procedure. This colour variation remains after tissue slides are digitalised as WSIs. Furthermore, the digital pathology techniques used for developing WSIs introduce illumination variation, which also results in colour variations [105]. The generalisation performance of CPath models is usually negatively affected by colour variation. Therefore, normalising the colour of different WSIs before subsequent analysis is important [56]. In the literature, various colour normalisation methods have been proposed and their effectiveness has been validated by extensive experiments, such as [75, 102].

Segmentation. Segmentation aims at assigning a label to each pixel inside images. The segmentation tasks of histopathology images generally include nuclei segmentation, mitosis detection and tissue type differentiation (gland, cancerous tissue, etc.). The detection, quantification and localisation of these components are crucial indicators for the diagnosis and prognosis of human cancers. In detail, morphological changes in the cell nucleus can provide clinically meaningful information during diagnosis, especially for cancers [18]. Tumour proliferation speed is an important biomarker for breast cancer, which is commonly assessed by counting mitotic figures in histological samples [104]. For colon cancer, the morphology of intestinal glands, including architectural appearance and glandular formation, is used by pathologists to inform prognosis and plan the treatment of individual patients [94]. The segmentation of tumour-infiltrating lymphocytes and characterising their spatial correlation on WSI is crucial in diagnosis, prognosis, and treatment response prediction for different cancers [26].

Diagnosis/Prognosis. The main challenge of applying algorithms on histopathology image-based cancer diagnosis and prognosis prediction is that the WSIs are of a large size (e.g. 100,000 100,000 pixels). It is infeasible to process the entire image at once by models, both due to the limitation of memory and a huge amount of computation. In practice, a diagnosis/prognosis prediction system usually adopts either patch-level based [15, 79, 119, 131] or WSI-level based [17, 92, 109, 122] methods. Patch-level based methods extract features from selected patches and use them for classification or prediction tasks. This kind of method can capture detailed local information; however, they do not fully utilise the spatial context and global information present in the entire WSI. In contrast, WSI-level based methods operate on the entire WSI, often at a lower resolution, to capture spatial relationships between different regions and structures from a global perspective.

In this thesis, we adopt existing colour normalisation methods for WSIs standardisation and focus on **segmentation** tasks. We leave the diagnosis/prognosis step for future work.

1.2 Research Objective and Contribution

The demand for a CAD system for pathologists to perform an objective, efficient, reproducible and quantitative analysis of a histopathology image has been put forward along with the emergence of digital pathology. To help improve diagnostic efficiency and to relieve pathologists from exhaustive and laborious manual examination, we aim to develop automated algorithms for the quantitative assessment of histopathology images. More specifically, we focus on two aspects, 1) the investigation of methods for the precise identification of regions within the tissue, which includes tumorous tissues, glands, and nuclei; 2) the development of methods that exploit rotational and scale symmetry within histopathology images to improve models' generalisation capability and robustness. Algorithms in the areas of machine learning, computer vision, and group representation theory will serve as the main tools for achieving the aims of this thesis. In line with the aforementioned aims, the following contributions have been made:

- We devise ADS_UNet, a stage-wise additive training algorithm that introduces deep supervision in hidden layers for highly discriminative feature learning, utilises cascade learning for efficient training and adopts AdaBoost for learning diverse features and model ensemble. The proposed ADS_UNet demonstrates state-of-the-art performance on four histopathology datasets and is much more computationally efficient.
- We propose Scale-Equivariant UNet (SEUNet), which is a UNet variant with filters built upon scale-space theory. The SEUNet contains groups of filters that are linear combinations of Gaussian basis filters, whose scale parameters are trainable but constrained to span disjoint scales through the layers of the network. Our proposed SEUNet demonstrates superior performance in generalising the trained model to unseen scales.
- We propose Rotation-Scale Equivariant Steerable Filters (RSESF), a filter formulation that jointly incorporates rotation equivariance and scale equivariance into CNNs to improve models' generalisation performance. The RSESF contains copies of filters, whose direction is controlled by directional derivatives and whose scale parameters are trainable but constrained to span disjoint scales in successive layers of the network. The proposed RSESF exhibits superior performance than other equivariant methods, with much fewer trainable parameters and fewer GPU resources required.

1.3 Structure of Thesis

Following this introductory chapter, the rest of the thesis is structured as follows:

Chapter 2 reviews the literature relevant to this study. It covers reviewing deep learning-based methods for histopathology image analysis, and scale/rotation equivariant neural networks.

Chapter 3 presents the proposed Adaboosted Deeply Supervised UNet (ADS_UNet), a stage-wise additive training algorithm that incorporates resource-efficient deep supervision in shallower layers and takes performance-weighted combinations of the sub-UNets to create the segmentation model. Empirical evidence on four histopathology datasets validates the effectiveness of ADS_UNet. Extensive ablation study and analysis are provided to reveal the reason for performance gain from the baseline model.

Chapter 4 presents the proposed Scale-Equivariant UNet (SEUNet), a UNet variant building on the scale-space theory, which demonstrates superior generalization performance on histopathology datasets at different scales when compared with the conventional CNN model and other scale-equivariant models.

In **Chapter 5**, we extend the SEUNet presented in **Chapter 4** to be a joint rotation and scale equivariant architecture, by utilising steerable filters. The Rotation-Scale Equivariant Steerable Filters (RSESF) is introduced in this chapter to improve the model's generalisation capability, by reducing its sensitivity to orientation and scale variation.

Chapter 6 summarises the findings and limitations of the work conducted in this thesis. We then propose future direction regarding computational histopathology.

1.4 Publications

1) **Yang, Yilong**, Srinandan Dasmahapatra, and Sasan Mahmoodi. "ADS_UNet: A nested UNet for histopathology image segmentation." Expert Systems with Applications (2023): 120128. **[Chapter 3]**

2) **Yang, Yilong**, Srinandan Dasmahapatra, and Sasan Mahmoodi. "Scale-Equivariant UNet for Histopathology Image Segmentation." In Geometric Deep Learning in Medical Image Analysis, pp. 130-148. PMLR, 2022. **[Chapter 4]**

3) **Yang, Yilong**, Srinandan Dasmahapatra, and Sasan Mahmoodi. "Rotation-Scale Equivariant Steerable Filters." In Medical Imaging with Deep Learning. PMRL, 2023 **[Chapter 5]**

Chapter 2

Literature review

Although attempts have been made to analyse WSIs based on machine learning algorithms to assist tasks including diagnosis, there are still very few commercial platforms of automated histopathology image analysis deployed in clinical use. With the rapid development of both technology and hardware, especially with the help of deep learning and Graphical Processing Unit (GPU), automated histology image analysis is eventually finding its way into clinical practice.

In this chapter, we review the literature related to deep learning-based histopathology image segmentation, with a particular focus on convolutional neural networks, as well as methods that help build models which are more robust and can be trained in a computationally efficient way. Firstly, some fully convolutional neural networks and transformer-based neural networks tasked with image segmentation and WSIs segmentation are reviewed. Secondly, approaches related to greedy training of deep neural networks that contribute to memory-efficient and computationally efficient training are reviewed. Then the boosting strategy (usually used to achieve better generalisation performance) and its cases of integration with deep neural networks are introduced. Finally, we address the fact that biopsy tissue can be captured at arbitrary orientation and magnification and result in cells appearing at different scales. Incorporating rotation and/or scale equivariance into neural networks may lower the orientation and scale sensitivity of models, thus leading to better generalisation. Therefore, literature related to building rotation and/or scale equivariance into neural networks is reviewed.

2.1 Deep Learning based Image Segmentation

Image segmentation is a key task in computer vision and image processing with important applications such as scene understanding, medical image analysis, autopilot, video surveillance, augmented reality, etc. Image segmentation can be formulated as

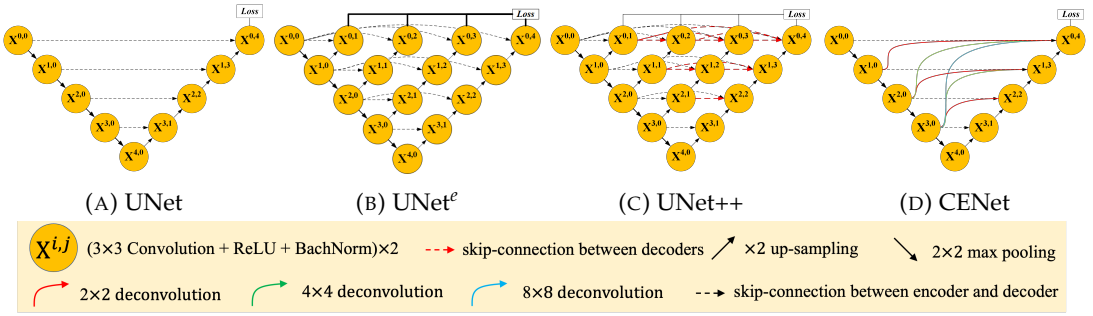


FIGURE 2.1: Comparison of (A) UNet, (B) UNet^e, (C) UNet++ and (D) CENet. UNet++ is constructed from UNet^e by introducing skip-connections (red dashed lines in (C)) between decoder nodes. CENet disregards inner decoder nodes and adopts deconvolution and concatenation to harvest multi-scale context clues between encoder and decoder nodes.

the problem of classifying pixels with semantic labels (e.g., normal tissue, tumour tissue, in the context of histopathology images). Before the advent of deep learning [63], most image segmentation algorithms were based on the combination of crafted feature extraction and machine learning algorithms. However, this field is currently dominated by methods based on deep neural networks. [80] reviewed more than 100 state-of-the-art deep learning based image segmentation algorithms and grouped them into 10 categories, and concluded that fully convolutional networks and encoder-decoder based models are two backbone models that are usually adopted and modified as components of other models. In 2015, [72] proposed the first end-to-end, pixel-to-pixel deep learning model which uses up-sampling layers to replace fully connected layers. This model is known as fully convolutional networks (FCN). The contribution of this work lies in the following aspects: 1) discarding fully connected layers reduces model complexity to a large extent. 2) using skip connections between the final layers of the model and feature maps of earlier layers to produce more accurate and detailed segmentation masks. However, because the encoder module reduces the resolution of the input by a factor of 32, the up-sampling layers struggle to produce a fine-grained segmentation mask.

UNet family. Building upon the concept of FCN, [88] proposed UNet that consists of a down-sampling path to capture context, and a symmetric up-sampling path to expand feature maps back to the input size. The down-sampling part has an FCN-like architecture that extracts features with 3 3 convolutions. The up-sampling part uses deconvolution to reduce the number of feature maps while increasing their area. Feature maps from the down-sampling part of the network are copied and concatenated to the up-sampling part to ensure precise localisation.

With the success of UNet, several variants have been proposed to further improve segmentation performance. Here we describe the networks UNet^e, UNet++ [128] and

CENet [126], whose simplified architectures are given in Figure 2.1. UNet^e is an ensemble architecture, which combines UNets of varying depths into one unified structure. Note that deep supervision [64] is required to train UNet^e in an end-to-end fashion. In order to allow deeper UNets to offer a supervision signal to the decoders of the shallower UNets in the ensemble and address the potential loss of information. The UNet++ connects the decoder nodes, to enable dense feature propagation along skip connections and thus more flexible feature fusion at the decoder nodes. The difference between UNet++ and UNet^e is that there are skip connections between decoder nodes in UNet++ (highlighted in red in Figure 2.1c). [126] proposed the contextual ensemble network (CENet), where the contextual cues are aggregated via densely upsampling the features of the encoder layers to the features of the decoder layers. This enables CENet to capture multi-scale context information. While UNet++ and CENet yield higher performance than UNet, they do so by introducing dense skip connections that result in a huge increase in parameters and computational cost. All of the methods mentioned above require the developer to manually pre-set the configuration of the network architecture (eg. the number and the size of the filter, the depth of the network, etc), the training-related parameters (eg. learning rate, weight decay, etc), and the pre-processing and post-processing strategies. This requires the end user to have high-level deep learning-related expert knowledge and thus greatly restricts their application in the medical image domain. To this end, [48] propose nnU-Net, a UNet-based segmentation method that automatically configures itself including preprocessing, network architecture, training and post-processing for any new task in the biomedical domain. The strong performance of nnU-Net is not achieved by a new network architecture, loss function or training scheme (hence the name nnU-Net, 'no new U-Net'), but by systematising the complex process of manual method configuration, which is previously addressed either by cumbersome manual tuning or purely empirical approaches with practical limitations.

Attention-based Architectures. Most recently, building upon the success of Vision Transformer [27] on image classification tasks, self-attention modules have also been integrated into UNet-like architectures for accurate segmentation. [73] proposed the hybrid ladder transformer (HyLT), in which the authors use bidirectional cross-attention bridges at multiple resolutions for the exchange of local and global features between the CNN- and transformer-encoding paths. The fusion of local and global features renders HyLT robust compared to other CNN-, transformer- and hybrid- methods for image perturbations. The comparison conducted on the Gland Segmentation (GlaS) dataset [94] shows that the HyLT improves the segmentation performance (mIoU score) by 13.46%, when compared with the UNet. [35] presented MedFormer, in which an efficient bidirectional multi-head attention (B-MHA) is proposed to eliminate redundant tokens and reduce the quadratic complexity of conventional self-attention to a linear level. Furthermore, the B-MHA liberates the constraints of model design and enables MedFormer to extract global relations on high-resolution token maps towards

the fine-grained boundary modelling. [74] proposed a hierarchical context-attention transformer-based architecture (HT-Net), which introduces an axial attention layer to model pixel dependencies of multi-scale feature maps, followed by a context-attention module that captures context information from adjacent encoder layers. Although the aforementioned models have achieved state-of-the-art performance in various medical image segmentation tasks, their training process requires significant GPU memory resources, which greatly limits the application of these models in resource-constrained platforms.

Deep Supervision. A deeply supervised network (DSN) [64] introduced classification outputs to hidden layers as well as the last layer output as is the convention. This was shown to increase the discriminative power of learned features in shallow layers and robustness to hyper-parameter choice. Despite the fact that the original DSN was proposed for classification tasks, deep supervision can also be used for image segmentation. [28] introduced deep supervision to combat potential optimisation difficulties and concluded that the model acquired a faster convergence rate and greater discriminability. Based on the UNet architecture, [129] introduced a supervision layer to each encoder/decoder block. Their method is very similar to our proposed supervision scheme (to be described in chapter 3); the difference lies in how the loss between the larger-sized ground truth and the smaller-sized output of hidden layers is calculated. Note that the dimensions of feature maps of the hidden layers are gradually reduced and become much smaller than that of the ground-truth mask, because of the down-sampling operation. In [28] and [129], deconvolutional layers were used to up-sample feature maps back to the same size as the ground-truth mask. Evidently, the additional deconvolutional layers introduce more parameters and more computational overhead. Although it was pointed out in [72] that one can learn arbitrary interpolation functions, bilinear interpolation was adopted in [118] to up-sample feature maps with no reduction in performance compared to learned deconvolutions. All of the aforementioned literature solve the dimension mismatch problem by up-sampling feature maps. Moreover, the authors assign balanced weights to the supervised layers, ignoring the fact that the importance of layers may be different throughout the training.

2.2 Application of CNNs on WSIs segmentation

Due to the large size of whole slide images, the general process of WSIs involves dividing the entire image into small patches, feeding these patches into the model to get output labels or masks and stitching them together to obtain the final segmentation mask of the entire image. Therefore, we transform the task of whole slide image segmentation to the segmentation/classification of image patches of this whole slide image. As a sub-field of image segmentation, automatic whole slide image segmentation also benefits a lot from the flourishing of deep learning. For example, all of top-10

segmentation solutions are deep learning based models, as reported in the technical reviews of the PAIP 2019 challenge [85].

[91] introduced and examined different strategies for the integration of multiple and widely separate spatial scales into common U-Net-based architectures and then proposed a family of end-to-end, multi-scale, multi-encoder fully-convolutional neural networks, which was termed msYI-Net. This model achieved 0.6596 mean IoU and ranked seventh on the leaderboard of the PAIP 2019 Challenge. The backbone of msYI-Net is UNet. [51] proposed an ensemble approach, which integrates DeepLabV3Plus [14] and UNets that are enhanced by DenseNet [46] and InceptionResNetV2 [100]. These three models were trained separately. The loss function used in all models is a linear combination of background dice loss, foreground dice loss and pixel-wise cross-entropy loss. The predicted probabilities of segmentation maps were averaged to generate the ensemble model prediction. Their approach achieved 0.7503 mean IoU and ranked third on the leaderboard of the PAIP 2019 Challenge.

The aforementioned two models resolve whole slide image segmentation by generating the segmentation mask of each patch, thus providing pixel-level classification. In contrast, the authors of [54, 106, 123], fed small image patches into models to get a class label for each image patch. In these approaches, all pixels of one patch share the same class label. So, it is a patch-level classification rather than a pixel-level classification. In most patch-level classification approaches, the spatial correlations among neighbouring patches are ignored since each image patch is predicted independently. This can result in inconsistent predictions over neighbouring patches. That is to say, there may be some isolated outliers after stitching all patches to form the whole segmentation mask. To tackle the issue, in [67], a neural conditional random field (NCRF) was proposed to model the correlation of neighbouring patches and directly incorporated it on top of a CNN feature extractor. Unlike the two-stage framework adopted in [54, 106, 123], this network can be trained end-to-end with a standard back-propagation algorithm. However, an explicit shortcoming of patch-level classification is that it doesn't consider the possibility of which boundary goes through an image patch.

2.3 Cascade/Greedy Training of Deep Neural Networks

The cascade learning strategy of artificial neural networks can date back to late 1980. [29] introduce the Cascade-Correlation learning architecture, which is originally proposed to solve the *moving target problem*, i.e., since all of the weights in a multi-layer perceptron are changing at once, each hidden unit sees a constantly changing environment. This, thus makes the learning of the neural network very slow. Different from end-to-end training where the topology of a network is fixed and all weights are adjusted at once, Cascade-Correlation begins with a small network and then gradually

adds and trains new hidden units. Once a new unit is added, all previous units are fixed and the network up to the newly added point acts as a permanent feature extractor. The key idea of Cascade-Correlation can break into two components: 1) *Cascade*, i.e., hidden units are appended into the network one at a time and are frozen after appending. 2) *Correlation*, the weight of the new hidden unit is determined by maximising the magnitude of the correlation between the new unit's output and the residual error signal. The strength of Cascade-Correlation can be summarised as follows: 1) Faster learning, as each unit sees a fixed problem and can move decisively to solve that problem. 2) Highly flexible architecture. There is no need to decide the size, depth, and connectivity pattern of the network in advance, as the model grows automatically.

Apart from growing the multi-layer perceptron (MLP) in a unit-wise fashion, gradually constructing and training neural networks in a coarser way (layer-wise [10, 44, 77], block-wise [45], module-wise [110]) has also been widely explored in the literature. In [44], the authors propose a two-stage method to greedily train a deep belief network one layer at a time. The first stage is unsupervised, i.e., training weights layer by layer to reconstruct the input data. For each layer, the outputs of the previous layer become the inputs of the next layer, with the weights of the previous layer being frozen. The second stage is supervised fine-tuning, i.e., all pre-trained weights of the entire network are trained to minimise the loss function by running gradient descent. Based on the work of [44], [10] demonstrated that greedy layer-wise unsupervised learning leads to better generalisation by initialising weights in an area near local minima. However, later works reveal that this initialisation is not necessary if specific network designs such as batch normalisation [47], skip connections [43], and dense connections [46] are adopted.

Another benefit of adopting cascade/greedy training of deep neural networks is a great saving in computation cost and GPU memory consumption. While end-to-end training usually suffers from a high GPU memory footprint, due to the need to store all intermediate activations for back-propagation. Motivated by the Cascade-Correlation idea of [29], [77] propose deep cascade learning for efficient training of deep neural networks in a bottom-up fashion. As shown in figure 2.2, the training process in cascade learning is progressively appending one layer to the existing network and only training the currently added layer and the corresponding output layer, whereas keeping the previous layers frozen. Experiments conducted in [77] demonstrate that cascade learning achieves competitive performance when compared with end-to-end training while showing a remarkable reduction in memory and time requirements. This is because all previously trained layers are not involved in the weight updating when training the newly added layer, thus there is no need to cache feature maps produced by those layers in GPU for back-propagation. Compared with [77] wherein every single layer is added to the network and then trained, [45] propose a telescoping sum technique

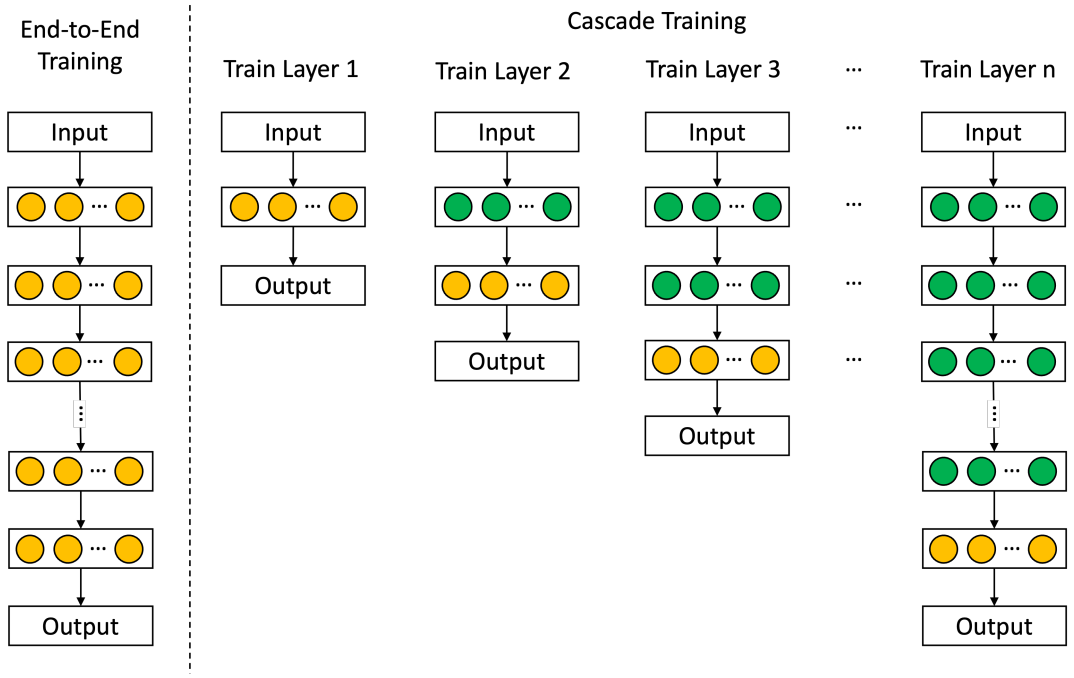


FIGURE 2.2: End-to-End training *v.s.* Cascade training. Yellow and green circles denote trainable weights and frozen weights, respectively. *Left.* The end-to-end training updates the parameters of the entire network after each round of back-propagation. *Right.* Cascade training starts from training a single-layer network to generate predictions, followed by appending layers to the top of the network gradually while freezing the previously trained layers. After all layers are fully trained, the output layers attached to the hidden layers are discarded and only the output layer of the last layer is kept as the final classifier.

to train a ResNet *block-by-block* (each residual block contains a mapping and an identity loop), while incorporating boosting theory for better representation learning. The resulting architecture is termed BoostResNet. Let M_1 and M_2 be the memory (or computation) required for one residual block and one linear classifier respectively, then the memory consumption and computation cost between BoostResNet and end-to-end trained ResNet is $M_1 \quad M_2$ *v.s.* $M_1 T \quad M_2$, where T is the number of residual blocks of the architecture.

Although cascade/greedy training demonstrates significant merit in terms of memory reduction and computational savings, it is worth mentioning that networks trained with local supervision usually show inferior performance than their end-to-end trained counterparts. The reason for this is an open-ended problem while an attempt [110] have been taken to explain and resolve this. For example, from the information-theoretic perspective, [110] believe that simply training local modules greedily with cross-entropy loss tends to collapse task-relevant information at earlier layers, resulting in the performance degradation of the full model. In other words, greedy training tends to be short-sighted and learns features that only benefit local modules, while ignoring the demands of the later layers. Based on this assumption, the authors propose a less greedy information propagation (InfoPro) loss that aims to encourage local modules

to propagate forward as much information from the inputs as possible, while progressively abandoning task-irrelevant parts. The authors experimentally demonstrate that the proposed InfoPro loss effectively prevents collapsing task-relevant information at local modules and thus yields better performance than other greedy training strategies, and even outperforms end-to-end trained architecture, while remaining the advantage of greedy training (fewer computational cost and GPU memory consumption).

2.4 Deep Ensemble Learning

The ensemble learning method is a class of machine learning methods that combine several individual models to reach better generalisation performance. Deep neural networks generally demonstrate a better feature extraction capability than traditional machine learning methods. Combining them results in deep ensemble learning models that aggregate the strength of both deep learning models and ensemble learning methods such that the final model exhibit powerful performance [33]. Depending on how the individual learners are generated, ensemble learning methods can be roughly grouped into two categories: 1) Bagging, and 2) Boosting [127]. In this subsection, we review literature that relates to integrating boosting strategies into deep neural networks.

AdaBoost. Boosting algorithms start with training a base learner and then update the weights of the training samples according to the result of the base learner such that incorrectly classified samples will receive more attention when training subsequent base learners. The boosting algorithm will finally combine many base learners into a single strong classifier whose prediction power is strong. As a representative boosting method, AdaBoost (Adaptive Boosting) [30] has been widely used in binary classification tasks. The idea of AdaBoost is based on the assumption that a highly accurate prediction rule can be obtained by combining many relatively weak and diverse rules. This was re-derived in [32] as a gradient of an exponential loss function of a stage-wise additive model. Such an additive model was extended to the multi-class case in [41], wherein the SAMME (Stage-wise Additive Modeling using a Multi-class Exponential loss function) is proposed to naturally extend the original AdaBoost algorithm to the multi-class case without reducing it to multiple two-class problems. The detailed iterative procedure of multi-class AdaBoost is described in Algorithm 1. Starting from equally weighted training samples, the AdaBoost trains a classifier f_t (t $1, 2, ..., T$ the iteration index) iteratively, re-weighting the training samples in the process. A misclassified item $\boldsymbol{x}_i$ is assigned higher weight $\boldsymbol{w}_i^t$ so that the next iteration of the training pays more attention to it. After each classifier f_t is trained, it is assigned a weight based on its error rate $_t$ on the training set. For the integrated output of the classifier ensemble, the more accurate classifier is assigned a larger weight $_t$ to have more impact on the final outcome. A classifier with $\frac{1}{C}$% accuracy (less than random guessing for C

Algorithm 1: Multi-class AdaBoost

Input: Training data: $\mathcal{X}$ $x_1, y_1, ..., x_m, y_m$; Number of weak classifiers: T; Number of classes: C.

w_i^0 $\frac{1}{m}, i$ $1, 2, ..., m$;
for t $1, 2, ..., T$ **do**
  h_t f_t $\mathcal{X}, w$;
  t $\frac{\sum_{i=1}^{m} w_i^t \; y_i \; f \; x_i}{\sum_{i=1}^{m} w_i^t}$;
  if t 1 $\frac{1}{C}$ **then**
    t $\frac{1}{2} \ln \frac{1 \; t}{t}$ $\ln \; C$ 1 ;
    w_i^t w_i^t 1 e^{t} $y_i \; f \; x_i$, i $1, 2, ..., m$;
    w_i^t $\frac{w_i^t}{\sum_{i=1}^{m} w_i^t}, i$ $1, 2, ..., m$;
  else
    t 0;
  end
end

Output: H $\arg\max_{C}$ $\sum_{t=1}^{T}$ t h^t

target classes) is discarded. T classifiers will be trained after repeating this procedure for T times. The final labels can be obtained by the weighted majority voting of these T classifiers.

DNNs with Boosting. Boosting has been widely adopted in the construction of deep neural networks. [24] propose AdaNet, a boosting-style algorithm that adaptively learns both the structure of the neural network and its weights. AdaNet uses the traditional boosting framework where weak classifiers are boosted. In AdaNet, features of hidden layers have to be fed into a classifier for ensembling, which leads to a very bushy structure of the model. Building upon the generalisation analysis of AdaNet, [45] explain the theoretical background for the success of deep residual neural networks (ResNet) [43] from the perspective of boosting theory. Based on the proposed *multi-channel telescoping sum boosting* framework, the authors introduce BoostResNet, an algorithm that trains ResNet block-by-block based on the principle of boosting. Different from traditional boosting that ensemble "estimated labels", BoostResNet boost over representations/features. In [101] the authors propose Adaboost-CNN, an adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning. In AdaBoost-CNN, all the weak classifiers are convolutional neural networks and have the same architecture. Instead of training a new CNN from scratch, they transfer the parameters of the prior CNN to the latter one and then train the new CNN for only one epoch, with the updated sample weights. This achieves better performance than the single CNN, but at the cost of increasing the number of parameters several folds.

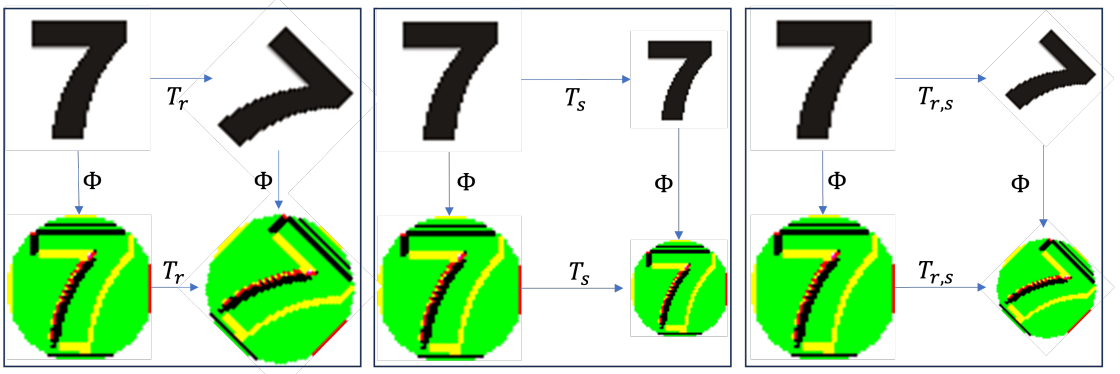
Curriculum learning [11] is related to boosting algorithms, in that the training schedule gradually emphasises the difficult examples. In [108], the authors show that a curriculum-based method can speed up the early layer training of cascade learning by using meaningful partitioned subsets. Similarly, [25] demonstrated that better performance can be achieved by forcing UNet to learn from easy to difficult scenes. However, the difficulty level of training samples is manually predefined according to some rules, rather than calculated by the network itself.

2.5 Equivariance in Convolutional Neural Networks

2.5.1 Symmetry, Equivariance, and Invariance

Symmetry. In the context of images, symmetry refers to a property of an object that remains unchanged under certain operations. There are various types of symmetrical patterns in images, such as translational symmetry, reflection symmetry, rotational symmetry, scale symmetry, etc.

Equivariance is a concept related to symmetry. Mathematically, equivariance is a form of symmetry for functions from one space with symmetry to another. In other words, a function is an equivariant map if its output changes in a predictable way when the input undergoes a specific transformation, and when the function commute with the transformation. Considering a function that maps an input space X to an output space Y and a transformation T acting on both X and Y, is equivariant with respect to T if $T\ x$ T x , for all x in X. Figure 2.3 show examples of equivariant mapping under rotation, scaling, and joint rotation-scaling transformations, respectively. Another example of equivariance is the built-in translation equivariance property of



(A) Rotation Equivariance (B) Scale Equivariance (C) Rotation-Scale Equivariance

FIGURE 2.3: From left to right, each sub-figures show that a mapping is equivariant under rotation transformation (T_r), scaling transformation (T_s), and joint rotation-scaling transformation (T_r, s), respectively. Where subscripts r and s denote a rotation angle and a scaling factor, respectively.

convolutional neural networks, i.e., by utilising spatial weight sharing strategy, it is guaranteed that a translation of the input will result in the same translation of the feature maps.

Invariance, on the other hand, refers to a property where the output of a function is not affected by the transform acting on the input. Mathematically, it means $T\ x$ x , x X, where T is the transformation and is a function. Invariance is important in image classification tasks where specific features or patterns of interest need to be recognised regardless of their exact position (translation-invariance), orientation (rotation-invariance), and scale (scale-invariance). For the consistency of segmentation, however, the equivariance property of a function is more important than invariance, as it is expected that the segmentation result of a transformed input should match the segmentation result of the original input.

One of the reasons that CNNs achieve remarkable success is owing to its translation equivariance. However, unlike translation equivariance, CNNs do not behave well in situations where large rotation and scale variations exist in the dataset, as those equivariant properties are not built into CNNs. To handle these variations (orientation, scale, etc.), most current methods usually make use of dataset augmentation, but this requires a larger number of model parameters and more training data, and results in significantly increased training time and a larger chance of under- or overfitting. The main reason for these drawbacks is that the learned model needs to capture adequate features for all the possible transformations of the input. Another pathway to deal with this variation is to encode rotation and scale equivariance into models, which have been explored either in a separate way or in a joint way.

2.5.2 Rotation Equivariant CNNs

There are numerous methods aimed at achieving rotation equivariance in CNNs. We categorise them into three classes as follows.

Rotating the data. The easiest, but most costly way to retain prediction variation introduced by orientation variation is by feeding the original input and its rotated copies into the model and then aggregating prediction to generate the final segmentation map. To this end, [81] propose test-time-augmentation (TTA). However, the inference time cost of TTA scales up linearly with the number of rotated copies. Training neural networks with data augmentation is widely used; however, there is no guarantee that CNNs trained with rotation augmentation will learn an equivariant representation and generalise to data with small rotations [4].

Rotating the filter. Instead of rotating the input, rotating the filter has also been widely explored. Cohen and Welling [20] propose group equivariant CNNs (G-CNNs), a generalisation of CNNs where rotation and reflection symmetries are embedded into G-convolution. G-convolution is built on the theory of *symmetry groups*. Different from conventional CNNs, the filter of G-convolution has an additional dimension (rotation dimension) where the rotated copies of the filter live in. The G-CNNs demonstrate state-of-the-art performance on classification tasks on rotated MNIST and CIFAR10 datasets, when compared with conventional CNNs. Based on the observation that histopathology images are inherently symmetric under rotation and reflection, the authors of [103] apply G-CNNs to the task of histopathology image segmentation and find that exploiting rotation equivariance improves tumour detection performance by 2.3 points (FROC score), on the Camelyon16 [7] dataset. This performance gap increases to 5.3 points in a small dataset regime wherein only 12.5% of data is used for training. However, the G-CNNs is limited to the discrete rotation group, and more specifically, the G-convolution proposed in [20] is limited to 90 rotations. To achieve a more fine-grained rotation equivariance, [9, 60] show that a rotation equivariant G-CNNs at arbitrary angular resolutions can be built by using bi-linear interpolation. They experiment on a series of G-CNNs equipped with different degrees of rotation equivariance (filter rotated by the step of $\frac{}{2}$, $\frac{}{4}$, $\frac{}{8}$) and demonstrate the effectiveness of the proposed interpolation method and the superiority of introducing higher degree of rotation equivariance, on tasks of histopathology image segmentation. In detail, the authors of [9] demonstrate that increasing angular resolution from $\frac{}{2}$ to $\frac{}{4}$ leads to 1.9 points improvement in F1-score, on the task of mitosis detection. However, the proposed interpolation method may introduce artefacts.

Steerable filters. Although restricting G-CNNs to discrete rotation equivariance leads to performance gain, encoding continuous rotation equivariance into CNNs will boost the performance further. In [21], the authors present a theoretical framework for understanding steerable representations in CNNs. The described mathematical theory can be applied to construct both discrete and continuous rotation equivariant filters. The benefit of utilising steerability is that it bypasses the issue of interpolation artefacts introduced by rotating filters by angles of non-90 . In [16], the authors propose to decompose the convolutional filters over a joint steerable bases across the space and the group geometry simultaneously, namely a rotation equivariant CNN with decomposed convolutional filters (RotDCF). [113] propose Steerable Filter CNNs (SFCNNs), which allows sampling an arbitrary number of filter orientations which improves the performance until saturation is reached. Experiments confirmed that SFCNNs generalise learned patterns over orientations and therefore achieve a lower sampling complexity than CNNs in rotation-equivariant recognition tasks. The authors of [115] introduce H-Nets (Harmonic Networks) that achieve continuous rotation equivariance by building filters out of the family of circular harmonics. When compared with a discrete rotation

equivariant G-CNN [20], the H-Nets reduces the classification error by 26% on the rotated MNIST dataset [61]. The authors of [112] extensively compare a wide range of steerable CNNs and release a PyTorch-based library[1] for equivariant deep learning.

2.5.3 Scale Equivariant CNNs

In the pre-deep learning era, Gaussian scale-space theory [68] was widely used for multi-scale image representation. Gaussian scale-space theory represents an image as a one-parameter family of gradually smoothed signals, in which the fine-scale details are successively suppressed by convolving the image with a set of re-scaled Gaussian filters and Gaussian derivative filters. Recently, [70] parameterised convolutional filters as a linear combination of Gaussian derivative filters with different scales, building neural networks robust to scale variations on image classification tasks. However, the architecture presented in [70] is only evaluated on image classification tasks, for which global scale invariance is key to predictive accuracy. For image segmentation tasks, the output map should scale in proportion to the input, making scale equivariance a necessary property. Similarly, [87] learn linear combinations of N^{th} order Gaussian derivative filters to create the N-Jet convolutional layer. Unlike [70] where the scale parameters ( ) are fixed, the and sizes of the filters in the N-Jet layer are learned from the data; this frees the network architect from searching and setting scale-related parameters for datasets and networks. However, the is shared by all filters in a layer, thus limiting the representational capacity of an N-Jet layer.

In recent years, group equivariance as an inductive bias for CNNs has influenced the design of several architectures including scale-equivariant convolutional networks. In [96], the authors propose Scale-Equivariant Steerable Networks (SESN), where filters are parameterised by a trainable linear combination of pre-calculated Hermite basis functions. Similarly, [130] propose Scale Decomposed Convolutional Filters (SDCF) that decompose the convolutional filters under two pre-determined separable bases and truncate the expansion to low-frequency components. [114] propose deep scale-space (DSS) based on the theory of scale-space and semi-groups to model transformation properties of images under scale transformations, modelling filter re-scaling by dilation. DSS exhibits 14.79 points higher mAP (mean Average Precision) than non-scale-equivariant CNN, on the segmentation task of the Cityscapes [23] dataset. However, the DSS is restricted only to integer scale factors and therefore does not cover a continuous range of scale variations. To extend DSS to arbitrary scales, the authors of [97, 98] propose Discrete Scale Convolution (DISCO) wherein the equivariance error between the non-integer scale factor with its two nearest integer scale factors is minimised. Compared with the DSS, the DISCO reduces the classification error by 37.32% on the STL-10 [19] dataset, with the help of introducing filters that are parameterised

[1] e2cnn: https://github.com/QUVA-Lab/e2cnn

with non-integer scale factors. All of the aforementioned methods define filters in the continuous scale domain and then project them onto pixel grids for a set of given scale factors. Although SESN and DISCO allow the use of arbitrary scale factors, the best set of scale factors are dataset and network-dependent and need to be carefully chosen to maximise model performance.

2.5.4 Rotation-Scale Equivariant CNNs

All of the literature mentioned in sections 2.5.2 and 2.5.3 encode the rotation and scale equivariance properties into CNNs separately. However, rotation and scale transformation usually simultaneously exist in real-world data, as images can be captured from any angle and the scale of objects depends on the distance between the camera and the scene.

Discrete $\mathcal{RST}$ equivariance. The $\mathcal{RST}$-CNN [34] is the first attempt to simultaneously incorporate $\mathcal{R}$otation, $\mathcal{S}$caling, and $\mathcal{T}$ranslation symmetries into CNNs. The $\mathcal{RST}$-CNN achieves discrete rotation and scale equivariance and is robust to nuisance data deformation. Numerical experiments on rotation and scale augmented MNIST [62], Fashion-MNIST [117], and STL-10 [19] datasets demonstrate that the proposed model yields remarkable gains over prior state-of-the-art equivariant methods, especially in the small data regime where both rotation and scaling variations are present within the data. In detail, the $\mathcal{RST}$-CNN boosts the classification accuracy on the STL-10 dataset by 13.7% and 23.38%, when compared with a rotation equivariant model (RESN [113]) and a scale equivariant model (SESN [96]), respectively. The performance gain soars up to 54.96% when compared with a non-equivariant CNN. However, the training of $\mathcal{RST}$-CNN requires the computation and GPU memory $N_r \quad N_s$ times more than what is needed by conventional networks. Where N_r and N_s are the numbers of rotation and scale channels of the filter, which greatly limits its applicability.

Continuous $\mathcal{RST}$ equivariance. Most recently, the Scale and Rotation Equivariant Network (SREN) [94] which achieves continuous equivariance with respect to both rotation and scaling is proposed. The underlying intuition behind the SREN consists of two steps: 1) obtaining the local scale and orientation of the image; 2) the captured information is used to adapt the scale and direction of the filter used for the convolution. By doing so, the filter is expected to transform as the input is transformed, thus the equivariance is guaranteed. To fulfil the first step, the authors propose the Scalable Fourier-Argand Representation to retrieve local geometric information and use it as a covariance indicator to determine the optimal orientation and scale through the calculation of the argmax. For step 2, the authors propose the Similarity Convolution (SimConv), a new convolution-like operation that satisfies the equivalence property. SERN does not suffer from the scaling of GPU requirements upon the increase of group size that other methods do. Experiments on the augmented MNIST and STL-10 datasets,

associated with mathematical analyses highlight the superior generalisation ability of the SREN on classification tasks when compared with other equivariant methods. The classification accuracy of the SREN on the STL-10 dataset surpasses that of the $\mathcal{RST}$-CNN by 5.11 points, owing to its continuous equivariance property.

2.6 Summary

In sections 2.1-2.4, we introduce literature on deep learning based image segmentation, cascade training and the integration of ensemble learning with neural networks. In chapter 3, we absorb cascade training and ensemble learning into the nested architecture of the UNet++, resulting in a more efficient and effective segmentation model. In section 2.5, we review the literature that aims at achieving rotation and scale equivariance in CNNs. Overall, CNNs equipped with a higher level of equivariance tend to show better resistance to orientation and scale variations and thus demonstrate better generalisation performance. However, making CNNs simultaneously rotation and scale equivariant has not been fully explored so far. Moreover, applying joint rotation-scale equivariant CNNs in the field of histopathology image segmentation is under-explored. In Chapter 4, we first construct a scale equivariant UNet (SEUNet), building upon scale-space theory. In chapter 5, we then extend SEUNet to be a joint rotation-scale equivariant CNN, with the help of steerable filters.

Chapter 3

Enhancing Segmentation via Ensemble Learning

3.1 Motivation and Contribution

The fully convolutional neural network (FCN) [72], trained end-to-end on per-pixel labels, is considered a milestone in image segmentation using deep networks. It was then extended by [88] to include a large number of up-sampled features concatenated using skip connections with the encoded convolutional features. They named the network a UNet after a geometrical layout of the network topology being a U-shape. [128] modified the UNet architecture by adding more nodes and connections to capture low-level correlation of distributed semantic attributes. The resulting architectures, known as UNet^e (e denotes ensemble) and UNet++, used class labels to guide the outputs of decoder layers (called deep supervision) to learn highly discriminative features.

Both UNet^e and UNet++ can be classified as ensemble models, in which multiple models are created to obtain better performance than each constituent model alone [82]. A property that is present in a good ensemble is the diversity of the predictions made by contributing models. However, end-to-end training of deep networks tends to correlate intermediate layers [49], hence the collaborative learning of constituent UNets adopted by UNet^e and UNet++ induces learned features to be correlated. Such learning runs counter to the idea of feature diversity pursued by ensemble models. Moreover, simple averaging performed in UNet^e, disregarding the difference in the performance of each member also restricts the final predictive performance of the ensemble.

Based on the work of UNet^e and UNet++, we pose several questions: 1) can each constituent model be forced to extract decorrelated features during training, to guarantee prediction diversity? 2) can the outputs of constituent models, sensitive to different

spatial resolutions, be weighted differently when they are integrated into the final segmentation? 3) can we provide deep supervision for encoders directly rather than by supervising the up-sampled decoders? To address these questions, we propose the Adaboosted Deeply Supervised UNet (ADS_UNet). The key contributions of our work can be summarised as follows:

1) We integrate deep supervision, cascade learning, and AdaBoost into the proposed ADS_UNet, a stage-wise additive training algorithm, in which multiple UNets of varying depths are trained sequentially to enhance the feature diversity of constituent models. Extensive experiments demonstrate that ADS_UNet is effective in boosting segmentation performance.

2) In our deep supervision scheme, we down-sample the mask to have the same size as feature maps of hidden layers to compute pixel-wise loss, instead of up-sampling features. This modification retains the advantages of deep supervision and yet reduces computation cost and GPU memory consumption.

3) Instead of assigning balanced weights to all supervised layers, we introduce a learnable weight for the loss of each supervised layer to characterise the importance of features learned by layers.

4) We conduct a comprehensive ablation study to systematically analyse the performance gain achieved by the ADS_UNet.

3.2 Method

Ensemble learning is often justified by the heuristic that each base learner might perform well on some data and less accurately on others for some learned features, to enable the ensemble to override common weaknesses. To this end, we seek enhanced segmentation performance of the model by enabling diverse feature maps to be learned. We propose the ADS_UNet algorithm, which adopts a block-wise cascade training approach [10, 29, 77] but with an added component that re-weights training samples to train each base learner in sequence. We evaluate the role of feature map diversity in section 3.5.3.

3.2.1 Computation and Memory Efficient Deep Supervision

As we mentioned in the introduction section, the UNet^e and UNet++ [128] offer deep supervision to shallower layers by gradually up-sampling feature maps to the size of the mask, which is computation and GPU memory expensive. To reduce the computational burden, we average-pool the mask to have the same size as feature maps. The

advantage of this change is that we no longer need to train deconvolutional weights for intermediate blocks to obtain feature maps with the same dimension as the ground-truth mask. We adopt UNet^{d}s, whose hidden layers have been trained with supervision, as base learners of the proposed ensemble model. Given the input image x and the network, we define the probability map generated at block $\mathrm{X}^{i,j}$ as:

$$\hat{y}^{i,j} \; x \qquad \text{softmax} \; \mathrm{X}^{i,j} \; x \tag{3.1}$$

The mapping $\mathrm{X}^{i,j} \quad : \boldsymbol{X} \qquad ^{N^{i,j} \;\; C}$ consists of a sequence of convolution, batch normalisation, ReLu activation and pooling operations, to transform the input image to a feature representation. Then a *softmax* activation function is used to map the representation to a probability map. Here C is the number of classes, $N^{i,j}$ denotes the number of pixels of the down-sampled mask, i, j denotes the index of convolutional blocks. Given mask $y^{i,j} \qquad ^{N^{i,j},C}$, the loss function used in the block $\mathrm{X}^{i,j}$ is the pixel-wise cross-entropy loss, which is defined as:

$$\mathcal{L}^{i,j} \; y^{i,j}, \hat{y}^{i,j}, N^{i,j} \qquad \frac{1}{N^{i,j}} \sum_{n \;\; 1}^{N^{i,j}} \sum_{c \;\; 1}^{C} y_{n,c}^{i,j} \log \; \hat{y}_{n,c}^{i,j} \;, \tag{3.2}$$

where $y_{n,c}^{i,j}$ is the ground-truth label of a pixel and $\hat{y}_{n,c}^{i,j}$ is the probability of the pixel being classified as class c. Based on equation 3.2, the overall loss function for the deep supervised UNet^d is then defined as the weighted sum of the cross entropy loss from each supervised block $\mathrm{X}^{i,j}$:

$$\mathcal{L}_d \qquad \sum_{i,j \;\; 0}^{i \;\; j \;\; d} \; {}_{i,j}^{d} \mathcal{L}^{i,j} \; y^{i,j}, \hat{y}^{i,j}, N^{i,j} \;, \quad {}_{d}^{i,j} \qquad 0, \qquad \sum_{i,j \;\; 0}^{i \;\; j \;\; d} \; {}_{d}^{i,j} \qquad 1, \tag{3.3}$$

where ${}_{d}^{i,j}$ is a weighting factor assigned to the convolutional block $\mathrm{X}^{i,j}$ to characterize the relative importance of blocks. d denotes the depth of the UNet. In contrast to previous works [28, 128, 129] that use equal weights ${}_{d}^{i,j} \qquad \frac{1}{d \;\; 1}$, we initialise ${}_{d}^{i,j}$ to $\frac{1}{d \;\; 1}$ and allow the ${}_{d}^{i,j}$ to be trainable, and use the *softmax* function to normalize ${}_{d}^{i,j}$ to guarantee ${}_{d}^{i,j} \qquad 1$. However, the feature learning of a block will be restricted if its ${}_{d}^{i,j}$ decreases to 0, during training. In order to guard against this competition exclusion phenomenon and encourage all supervised blocks to contribute to the segmentation, we add a constant $\frac{1}{d \;\; 1}$ to ${}_{d}^{i,j}$ to raise its lower limit:

$$\tilde{}_{d}^{i,j} \qquad \frac{{}_{d}^{i,j} \qquad \frac{1}{d \;\; 1}}{\sum_{i,j \;\; 0}^{i \;\; j \;\; d} \; {}_{d}^{i,j} \qquad \frac{1}{d \;\; 1}} \qquad \frac{{}_{d}^{i,j}}{2} \qquad \frac{1}{2 \;\; d \;\; 1}, \tag{3.4}$$

Since $\lim_{\substack{i,j \\ d \ 0}} \frac{\tilde{}^{i,j}_{d}}{2} \quad \frac{1}{2\ d\ \ 1} \quad \frac{1}{2\ d\ \ 1}$ and $\lim_{\substack{i,j \\ d \ 1}} \frac{\tilde{}^{i,j}_{d}}{2} \quad \frac{1}{2\ d\ \ 1} \quad \frac{d\ \ 2}{2\ d\ \ 1}$, $\tilde{}^{i,j}_{d}$ is bounded in $\frac{1}{2\ d\ \ 1}, \frac{d\ \ 2}{2\ d\ \ 1}$.
Then equation (3.3) is re-written as follows to train each constitute model UNet^d:

$$\begin{aligned} &\mathcal{L}_d \sum_{\substack{i,j \ 0}}^{i\ \ j\ \ d} \tilde{}^{i,j}_{d} \mathcal{L}^{i,j}\ y^{i,j}, \hat{y}^{i,j}, N^{i,j}\ , \\ &\frac{1}{2\ d\ \ 1} \quad \tilde{}^{i,j}_{d} \quad \frac{d\ \ 2}{2\ d\ \ 1}, \sum_{\substack{i,j \ 0}}^{i\ \ j\ \ d} \tilde{}^{i,j}_{d} \quad 1. \end{aligned} \tag{3.5}$$

Once the UNet^d is trained, the final probability map generated by UNet^d is calculated by:

$$\hat{y}_d\ x \sum_{\substack{i,j \ 0}}^{i\ \ j\ \ d} \tilde{}^{i,j}_{d} \hat{y}^{i,j}\ x\ , \tag{3.6}$$

with $\hat{y}^{i,j}\ x$ and $\tilde{}^{i,j}_{d}$ defined in equations (3.1) and (3.4). $\hat{y}_d\ x$ denotes the combined prediction of model UNet^d. We conduct ablation studies in section 3.5.2 to show the benefits of imposing range constraint on $\tilde{}^{i,j}_{d}$. Moreover, we demonstrate that generating the final prediction by using the weighted summation of multi-scale outputs yields better segmentation performance.

3.2.2 Stage-wise Additive Training

The stage-wise additive training process of the ADS_UNet is described in Algorithm 2 and visually illustrated in Figure 3.1. The main components of the iterative training procedure are 1) updating sample weights, 2) assigning weighting factors to base learners, and 3) freezing trained encoders while training decoders. We will elaborate on these as follows.

Firstly, given the training images $\mathcal{X}= x_1, ..., x_m$ of n pixels each, and associated masks $\mathcal{Y}= y_1, ..., y_m$, we assign a weight w_k to each sample x_k. These weights are initialised to $w_k^1 \quad \frac{1}{m}$ (line 1 in Algorithm 2). Then, in the first iteration (d=1), the parameters of the encoder block (X^{0,0}) of the first base learner UNet^1 are initialised (line 2). In the first iteration of the sequential learning approach, parameters of the bottleneck node X^{1,0} and decoder nodes X^{0,1} of the UNet^1 are initialised randomly (lines 4-6). Line 7 initialises the weighting factors $\tilde{}^{i,j}_{d}$ of supervised blocks. The UNet^1 is then trained on all training samples with the same weight of $\frac{1}{m}$ (line 8). After the UNet^1 is trained, the training set will be used to evaluate it and to determine its error rate $_1$ (lines 9-11). In contrast to AdaBoost, we use mean Intersection over Union (mIoU) error (lines 10) to measure segmentation performance rather than using a misclassification rate. In detail, given one-hot mask $\boldsymbol{y}_{k,c}= k_1, \quad , k_n\ , k_j \quad 0,1$ for a pixel of image k belonging to class c and the corresponding one-hot prediction $\hat{\boldsymbol{y}}^d_{k,c}= \hat{k}_1, \quad , \hat{k}_n\ , \hat{k}_j \quad 0,1$ generated

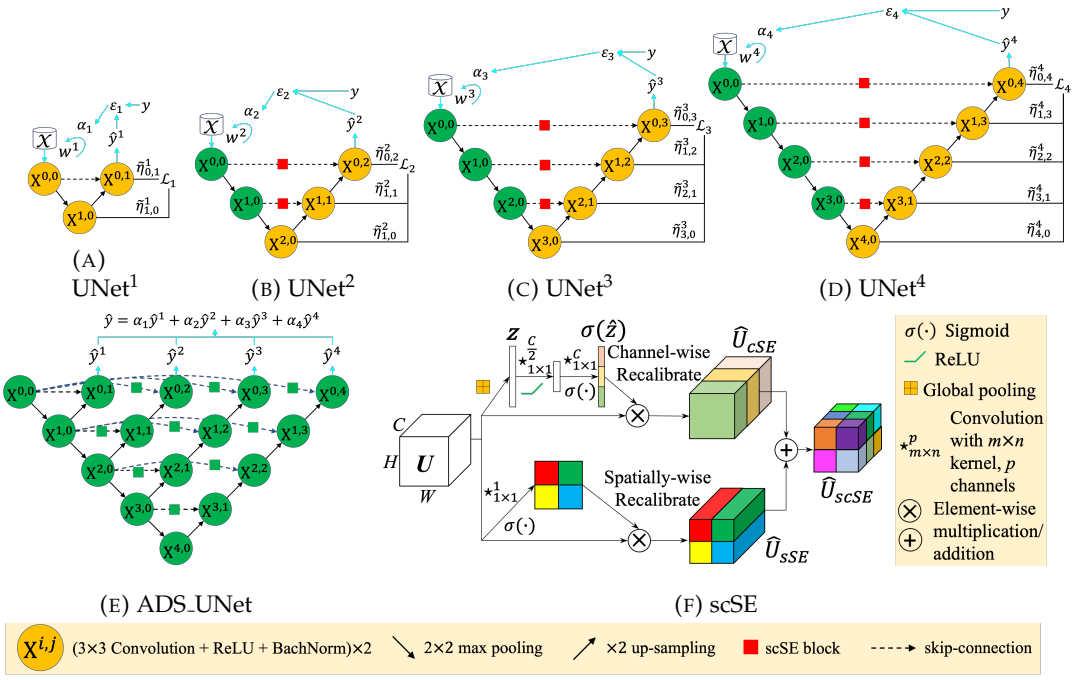


FIGURE 3.1: The architecture of the proposed ADS_UNet. Each circular node in the graph represents a convolution block. Specifically, yellow nodes indicate that parameters are trainable during back-propagation, and green nodes indicate that parameters are frozen. (a-d) UNets of varying depths. All of UNet^i are trained with the same dataset $\mathcal{X}$, but using different sample weights, W^i. (e) Ensemble architecture, ADS_UNet, which combines UNets of varying depths into one unified architecture for inference. (f) The scSE block. It contains *bottom branch* channel squeeze and spatial excitation block (sSE), and *up branch* spatial squeeze and channel excitation block (cSE).

by UNet^d, the mIoU score s_k^d is calculated by:

$$s_k^d \quad \text{mIoU} \; y_k, \hat{y}_k^d \quad \frac{1}{C} \; {}_c \; {}_1 \; \frac{C \quad y_{k,c} \quad \hat{y}_{k,c}^d}{y_{k,c} \quad y_{k,c} \quad \hat{y}_{k,c}^d \quad \hat{y}_{k,c}^d \quad y_{k,c} \quad \hat{y}_{k,c}^d}, \tag{3.7}$$

where k is the index of training images, c is the index of class labels, d is the index of iteration and also denotes the depth of the constituent UNet. If the error rate $\;_1$ of the UNet^1 is less than 1-$\frac{1}{C}$ (line 12), then UNet^1 will be preserved for the ensemble, otherwise, it will be disregarded by setting its weighting factor to 0 (lines 18-19). In the case that $\;_1 \quad 1 \quad \frac{1}{C}$, the equation shown in line 13 is used to calculate model weight $\;_d$ for the ensemble. So far we have obtained the first base learner UNet^1, and its weighting factor $\;_1$.

Algorithm 2: ADS_UNet. The $\tilde{}_{i,j}^{t}$ term in line 9 is discussed in the context of equation (3.4); the UNet^t are described in Figure 3.1.

Input: Number of class: C; Training images: $\mathcal{X}$ $x_1, ..., x_m$; Training masks: $\mathcal{Y}$ $y_1, ..., y_m$; Number of iteration: T.

1 $\boldsymbol{w}^1$ w_k^1 w_k^1 $\frac{1}{m}, k$ $1, 2, ..., m$;
2 Initialising convolutional block $X^{0,0}$;
3 **for** d $1, 2, ..., T$ **do**
4 **for** j $0, 1, ..., d$ **do**
5 Initializing convolutional block $X^{d\ j,j}$;
6 **end**
7 $\tilde{}_{d}^{i,j}$ $\frac{1}{d\ 1}$, i, j 0, i j d ;
8 Train UNet^d $\mathcal{X}, \mathcal{Y}, \boldsymbol{w}^d$;
9 $\hat{y}_k^d$ $_{i,j\ 0}^{i\ j\ d}$ $\tilde{}_{d}^{i,j} \hat{y}^{i,j}$ x_k ; (3.6)
10 s_k^d mIoU $\hat{y}_k^d, y_k$; (3.7)
11 $_d$ $_{k\ 1}^{m}$ w_k^d 1 s_k^d ;
12 **if** $_d$ 1 $\frac{1}{C}$ **then**
13 $_d$ $\frac{1}{2} \ln \frac{1\ _d}{_d}$ $\ln C$ 1 ;
14 Updating sample weight w_k^d using equation (3.8a) and (3.8b);
15 **for** j $0, 1, ..., d$ **do**
16 Freeze convolution block $X^{j,0}$;
17 **end**
18 **else**
19 $_d$ 0 ;
20 **end**
21 **end**

Output: ADS_UNet $\arg\max_{C}$ $_{d\ 1}^{T}$ $_d \hat{y}_d$

We then update sample weights based on mIoU scores (line 14) for the training of the next iteration:

$$w_k^d \quad w_k^d\ ^1 e^{\ 1\ s_k^d\ 1}, \quad k \quad 1, 2, ..., m, \tag{3.8a}$$

$$w_k^d \quad \frac{w_k^d}{_{i\ 1}^{m} w_i^d}, \quad k \quad 1, 2, ..., m, \tag{3.8b}$$

Equation (3.8a) assigns greater weight to images that cannot be accurately segmented by UNet$^{d\ 1}$, encouraging UNet^d to focus more on their segmentation. Equation (3.8b) normalizes sample weights to guarantee that $_{k\ 1}^{m} w_k^d$ 1.

Before the start of the second iteration, it is necessary to freeze the encoder nodes ($X^{0,0}$ and $X^{1,0}$) of the UNet^1 (lines 15-17). Otherwise, the process of training UNet^2 would update UNet^1's encoder parameters as well, reducing the learned association between the encoder and decoder paths of UNet^1. Furthermore, subsequent sub-networks UNet^d, for d 2 would acquire correlated features. The code block in lines 4-20 is run for T iterations to obtain T base learners, each weighted by $_d$. Note that all parameters

of UNet[1] are trained as a whole but UNet[2] reuses encoder weights of UNet[1] and only its decoder parameters are trained (if $_1$ 1 $\frac{1}{C}$). These are shown as yellow nodes in Figure 3.1b. The purpose of using the updated sample weights w_k^2 to train UNet[2] is to force the decoder layers of UNet[2] (because the connection between $X^{1,0}$ and $X^{2,0}$ only involves max pooling) to learn features dissimilar to those learned by UNet[1]. This procedure is repeated for each of the base learners UNet^d, with the additional help of feature normalisation to be described next.

3.2.3 Feature Re-calibration

The concurrent spatial and channel Squeeze & Excitation (scSE) block [89] is used to re-calibrate feature maps learned from encoder blocks of UNet^d, to better adapt to features learned from decoder blocks of deeper UNet$^{d\ a}$, a 1 layers. For example, features learned by the encoder block $X^{0,0}$ and the decoder block $X^{1,0}$ of the UNet[1] can cooperate well to perform segmentation since their weights are updated in a coordinated end-to-end back-propagation process. In UNet[2], however, features produced by $X^{1,1}$ and $X^{1,0}$ (in the same depth) can be very different, since the gradient flow is truncated between block $X^{1,0}$ and $X^{2,0}$. Therefore, although features produced by $X^{0,0}$ used to cooperate well with that of $X^{1,0}$, it is not guaranteed that it can adapt well to that of $X^{1,1}$. Based on this analysis, the scSE block is used to re-weight features before concatenating. We evaluate the role of feature re-calibration in section 3.4. The detailed process of scSE is illustrated in Figure 3.1f.

Given an input feature map $\boldsymbol{U}$ $^{H\ W\ C}$, The channel squeeze operation generates a matrix $\boldsymbol{q}$ $^{H\ W}$ with matrix elements $q_{i,j}$ W_{sq} $U_{i,j,k}$, W_{sq} ^{C} maps the vector at each location i, j into a scalar. This matrix is then re-scaled by passing it through a sigmoid function , which re-weights the input feature map $\boldsymbol{U}$ spatially,

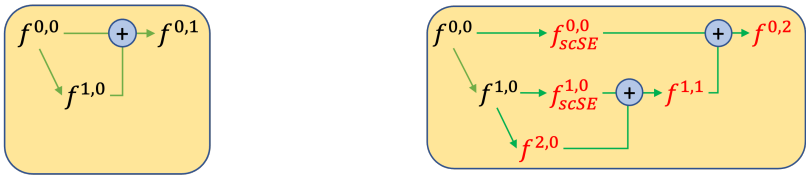
$$\hat{U}_{i,j,k}^{sSE} \quad q_{i,j}\ U_{i,j}, \tag{3.9}$$

The global average pooling of the feature map over all pixels produces $\boldsymbol{z}$ with components z_k,

$$z_k \quad \frac{1}{H \quad W} \sum_{i}^{H} \sum_{j}^{W} U_{i,j,k}, \quad k \quad 1, 2, \ldots, C \tag{3.10}$$

This vector, $\boldsymbol{z}$, is transformed to $\hat{\boldsymbol{z}}$ $\boldsymbol{W_1}$ ReLU W_2z , with $\boldsymbol{W}_1$ $^{C\ \frac{C}{2}}$, $\boldsymbol{W}_2$ $^{\frac{C}{2}\ C}$ being weights of two fully connected layers. The range of the activations of $\hat{\boldsymbol{z}}$ are brought to the interval [0, 1], by passing it through a sigmoid function . The input feature map $\boldsymbol{U}$ is then re-weighted by the re-scaled vector, with its k^{th} channel

$$\hat{\boldsymbol{U}}_k^{cSE} \quad \hat{\boldsymbol{z}}_k\ \boldsymbol{U}_k, \boldsymbol{U}_k \quad ^{H\ W}. \tag{3.11}$$



(A) Info flow of UNet^1 (B) Info flow of UNet^2

FIGURE 3.2: Information flow diagram of base learners. $f^{i,j}$ denotes the output feature maps of block $X^{i,j}$, $f^{i,j}_{scSE}$ denotes the re-calibrated version of $f^{i,j}$, a circle with plus sign denotes feature map concatenation operation, connecting lines with an arrow denote the flow of features. Showing in red features that can be updated during the training of UNet^2 (after UNet^1 is trained), while others are fixed.

In the channel re-calibrated feature maps $\hat{\boldsymbol{U}}^{cSE}$, the channels that are less important are suppressed and the important ones are emphasized. Finally, after concurrent spatial and channel squeeze and excitation (scSE), a location i, j, c of the input feature map $\boldsymbol{U}$ is then given higher activation when it gets high importance from both, channel re-scaling and spatial re-scaling.

It is worth mentioning that sample re-weighting and feature re-calibration are utilized in the ADS_UNet for different purposes and are not in conflict with each other. Taking the pair UNet^1 and UNet^2 as an example, sample re-weighting aims at achieving feature diversity between final outputs ($f^{0,1}$ and $f^{0,2}$ in Figure 3.2) of two base learners, so that the ensemble of UNet^1 and UNet^2 can compensate for each other's incorrect predictions, thus leading to better segmentation. When considering feature re-calibration, UNet^1 is trained as a whole with each training sample having the same sample weight (as described in section 3.2.2). That means feature maps $f^{0,0}$ and $f^{1,0}$ have a high association. In the second iteration, however, UNet^2 reuses UNet^1's encoder blocks ($X^{0,0}$ and $X^{1,0}$ are fixed now), only newly added blocks ($X^{2,0}$, $X^{1,1}$ and $X^{0,2}$) are trained on updated sample weights. This reduces the association of $f^{1,0}, f^{2,0}$ and $f^{0,0}, f^{1,1}$, enabling feature de-correlation between fixed and newly added feature maps. Directly concatenating $f^{1,0}$ with $f^{2,0}$ and $f^{1,1}$ with $f^{0,0}$ ignores the feature dependence issue and results in lower performance (validated in Table 3.4). Therefore, to mitigate this feature mismatching effect, we re-calibrate the fixed features ($f^{0,0}$ $f^{0,0}_{scSE}$, $f^{1,0}$ $f^{1,0}_{scSE}$), before concatenation.

3.2.4 Difference between ADS_UNet and UNet++

In section 3.2.1- 3.2.3, we introduced the components and training scheme of the ADS_UNet. For inference, the final probability map for an image x $^{C\ H\ W}$ can be generated by weighted average:

$$\hat{y}\ x \quad \text{ADS\_UNet}\ x \quad {}^{T}_{d\ 1}\ {}_d\hat{y}_d\ x \tag{3.12}$$

Here C is the number of classes. $\hat{y}_d\ x$ is the probability map generated by UNet^d, as defined in equation (3.6) and shown in Figure 3.1e.

The proposed ensemble structure differs from the UNet++ in two ways: one differs in the training method, and the other in the way decisions are made and incorporated into learning. 1) *Embedded vs. isolated training*. The UNet++ is trained in an embedded training fashion where the full UNet++ model is trained as a whole, with deep supervision on the last decoder block $X^{0,i}$ of branch i. In the ADS_UNet, however, each UNet^d is trained by isolating features acquired by the deeper encoder and decoder blocks. Moreover, deep supervision is added to each decoder block of each branch by down-scaling the label masks, rather than solely on the last decoder node of each branch. 2) *Average vs. weighted average voting.* In the ensemble mode of the UNet++, the segmentation results from all branches are collected and then averaged to produce the final prediction. UNet++ x $\arg\max_{c\ C} \frac{1}{T}\ _{d\ 1}^{T}$ UNet^d x , with UNet^d x $\hat{y}^{0,d}$. T is the number of branches of the UNet++. However, the ADS_UNet takes performance-weighted combinations of the component UNets to create the final segmentation map: ADS_UNet x $\arg\max_{c\ C}\ _{d\ 1}^{T}\ _d\hat{y}_d$, with $\hat{y}_d$ UNet^d x is calculated from equation (3.6). $_d$ reflects the importance of the UNet^d in the ensemble.

3.3 Experiments and Results

Four histopathology datasets are used to check the effectiveness of the proposed methods.

3.3.1 Datasets

CRAG dataset. The colorectal adenocarcinoma gland (CRAG) dataset [3] contains a total of 213 Hematoxylin and Eosin images taken from 38 WSIs scanned with an Omnyx VL120 scanner under 20× objective magnification). All images are mostly of size 1512 1516 pixels. The dataset is split into 173 training images and 40 test images. We resize each image to a resolution of 1024 1024 and then crop it into four patches with a resolution of 512 512 for all our experiments (852 patches in total).

GlaS dataset The Gland Segmentation dataset [94] contains a total of 165 images (20× objective magnification) which are originally split into 85 images for training and 80 for testing. We crop four corners with the size of 512×512 from each image, resulting in 660 patches in total.

BCSS dataset. The Breast Cancer Semantic Segmentation dataset [2] consists of 151 H&E stained whole-slide images and ground truth masks corresponding to 151 histologically confirmed breast cancer cases. A representative region of interest (ROI) was selected within each slide by the study coordinator, a medical doctor, and approved by a senior pathologist. ROIs were selected to be representative of predominant region classes and textures within each slide. Tissue types of the BCSS dataset consist of 5 classes (i)tumour, (ii)stroma, (iii)inflammatory infiltration, (iv)necrosis and (v)others. We slide a 512 512 window over ROIs with the stride of 256 to crop patches from training and test images, resulting in 4376 image tiles in total. Since the class label of pixels is quite imbalanced, a weighted categorical cross-entropy loss is used to mitigate class imbalance, with the weight associated with each class determined by W_c 1 $\frac{N_c}{N}$, where N is the number of pixels in the training dataset and N_c is the number of pixels belonging to class c.

MoNuSeg dataset. The MoNuSeg dataset [58] is a multi-organ nucleus segmentation dataset. The training set includes 37 images of size 1000 1000 from 4 different organs (lung, prostate, kidney, and breast). The test set contains 14 images with more than 7000 nucleus boundary annotations. A 400 400 window is used to slide through the images with a stride of 200 pixels to separate each image into 16 tiles.

5-fold cross-validation. For each dataset, we gather the image patches cropped from both training and testing images together and then randomly divide them into 5 subsets for cross-validation. 5-fold cross-validation strikes a balance between computational efficiency and obtaining a reliable estimate of the model's performance. Increasing folds from 5 to 10 will reduce the risk of over-fitting (especially in a small data regime), thus reducing the variability in performance estimates and providing a potentially more accurate reflection of the model's generalisation performance. However, 10-fold cross-validation requires more iterations, leading to a higher computational cost compared to 5-fold cross-validation.

3.3.2 Baselines and Implementation

Since our work is mainly based on UNet, UNet^e, and UNet++, we re-implement these three models, as well as CENet, to compare with our proposed methods. We also compare the proposed ADS_UNet with two transformer-based UNet variants, HyLT [73] and MedFormer [35], using the implementation provided by the authors. For a fair comparison, the configuration of the outermost convolutional blocks ($X^{i,0}$, i $0,1,2,3$ and $X^{i,j}$, i,j 0, i j 4) of all compared methods are exactly the same as in the original UNet (both the number and size of filters). All inner decoder nodes of UNet^e, UNet++ and ADS_UNet are also exactly the same, and all models have the same hyper-parameters. It is noted that scSE block is not used in UNet, UNet^e, UNet++

and CENet, but it is used in the skip-connections of ADS_UNet. The models are implemented in Pytorch [86] and trained on one NVIDIA RTX 8000 GPU using the Adam optimizer [52] with weight decay of 10^{-7} and learning rate initialised at 0.001 and then changed according to the 1cycle learning rate policy [95]. The cross-entropy loss is used to train all compared models, and ADS_UNet is trained with the linear combination of loss functions using equation (3.5). On models with a depth of 4, the number of filters at each level are 64, 128, 256, 512, and 1024, on the CRAG, GlaS, and BCSS datasets. This setting is consistent with the standard UNet [88]. However, we change the number of filters to 16, 32, 64, 128, 256 for all models, when trained on the MoNuSeg dataset, as our experimental results show that increasing the number of filters leads to inferior performance. The colour normalisation method proposed in [102] is used to remove stain color variation, before training. We also compare our methods with the state-of-the-art nnU-Net [48]. Note that the nnU-Net automatically decides the depth of the architecture based on its characterisation of the properties of the datasets. In our experiments, the nnU-Net generated for the MoNuSeg dataset is 6, while it is 7 for the GlaS, CRAG, and BCSS datasets. The officially released nnU-Net source code is used in our experiments.

Net	CRAG		BCSS		MoNuSeg		GlaS	
UNet [88]	84.38	0.95	71.22	1.43	86.07	0.58	91.49	0.95
UNet^e [128]	84.60	0.74	73.20	0.84	87.07	0.32	92.16	0.68
UNet++ [128]	84.34	0.91	72.35	0.39	87.13	0.38	92.07	0.73
nnU-Net [48]	**87.88**	**1.06**	67.94	1.16	83.55	0.40	**92.87**	**0.75**
CENet [126]	82.92	1.68	69.06	0.92	86.55	0.44	91.29	0.98
HyLT [73]	86.08	0.98	69.42	1.16	85.62	0.47	88.54	0.35
MedFormer [35]	85.23	0.87	68.64	1.40	83.54	0.39	89.95	0.50
ADS_UNet	86.92	0.88	**75.73**	**0.60**	**87.78**	**0.41**	92.65	0.71

TABLE 3.1: Segmentation results (mIoU std) of UNet variants and ADS_UNet. The boldface highlights the highest score.

3.3.3 Results

Some image patches and their corresponding segmentation maps are depicted in Figure 3.4. Table 3.1 summarises the segmentation performance achieved by all compared methods. The performance of the baseline method (VGG-16, FCN-8) used in [2] is also included for comparison. The number of parameters and computational efficiency of various UNet variants is reported in Table 3.2.

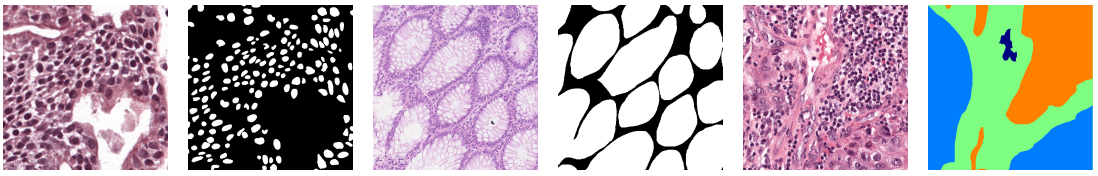
Among the different networks evaluated, the ADS_UNet outperforms all of the other state-of-the-art approaches on the BCSS and MoNuSeg datasets, achieving competitive performance on the CRAG and GlaS datasets. UNet++ achieves 1.13, 1.06 and 0.58 higher mIoU scores than UNet on BCSS, MoNuSeg and GlaS datasets by performing

Net	Params (M)	FLOPs (G)	GPU (GB)	Time (s)
UNet [88]	31.04	218.9	5.54	771
UNet^e [128]	34.92	445.2	9.8	1071
UNet++ [128]	36.17	514.8	9.31	1303
nnU-Net [48]	41.27	65.6	2.92	442
CENet [126]	35.17	471.55	5.99	713
HyLT [73]	42.2	329.11	16.06	1500
MedFormer [35]	99.54	325.76	15.48	1337
ADS_UNet	0.41 1.63 6.65 26.72	62.61 114.80 166.93 219.04	4.00 4.92 5.40 5.71	453

TABLE 3.2: The comparison of models in terms of the number of parameters, computational complexity (measured by FLOPs), required GPU memory, and training time (seconds) per epoch. The FLOPs and GPU consumption are computed with 512 512 inputs. The GPU memory consumption is measured by *nvidia-smi* command (batch size=2). In ADS_UNet, base learners require a different amount of GPU memory, since they vary in depth and the number of parameters (The total number of trainable parameters of the ADS_UNet is 35.41 million). These statistics are measured during the training on the BCSS dataset.

2.35 times more computation and consuming 1.77 times more GPU memory. In contrast, ADS_UNet outperforms the UNet++ and yet requires at most 59.51% of the GPU memory and 42.55% of the floating-point operations required by UNet++ for training.

nnU-Net slightly outperforms the ADS_UNet on the GRAG and GlaS datasets but is much weaker than ADS_UNet on the BCSS (7.79) and MoNuSeg (4.23) datasets. The design choices (pipeline fingerprint) of nnU-Net are not fixed across datasets but are configured on the fly according to the 'data fingerprint' (dataset properties such as image size, image spacing, number of classes, etc.). The data-dependent 'rule-based parameters' (patch size, batch size, network depth, etc.) of the pipeline is determined by a set of heuristic rules that models parameter inter-dependencies. Therefore, the performance of nnUNet can be explained as follows: Firstly, the nnU-Net is deeper (the depth of the nnU-Net is 6 or 7, as mentioned in section 3.3.2), which means that the convolutional kernels of the bottleneck layer (the deepest encoder layer) have a larger receptive field, enabling the model to extract information from a larger region. This is especially beneficial when the task is to recognise large objects (e.g. glands) since a larger receptive field can cover the whole object. In models with a depth of 4, the size of the receptive field of the bottleneck layer is limited. This difference in the depth of models may explain why nnU-Net outperforms shallower models when trained for segmenting glands. In contrast, the size of the cell nucleus in the MoNuSeg dataset is much smaller than glands in the CRAG and GlaS datasets. In this case, the receptive field of the bottleneck layer of shallow models is large enough to capture the entire nucleus. Further increasing the depth of the network compresses features leading to information loss rather than enhancing the features learnt. Therefore, we argue that the nnU-Net improves segmentation performance by enlarging receptive field size, while ADS_UNet achieves so by ensembling. Image and mask patches presented in Figure 3.3



(A) (B) (C) (D) (E) (F)

FIGURE 3.3: (A)-(B) Image-mask patch from the MoNuSeg dataset (nucleus segmentation). (C)-(D) Image-mask patch from the CRAG dataset (gland segmentation). (E)-(F) Image-mask patch from the BCSS dataset (breast cancer segmentation). All patches are of size 512 512.

show the size difference of target objects between the nucleus segmentation dataset and tissue type (gland, tumour) segmentation dataset.

Both transformer-based architectures, HyLT and MedFormer, demonstrate inferior performance on all four datasets. Moreover, it is worth noting that the HyLT and the MedFormer have 1.19 times and 2.81 times parameters than the ADS_UNet does and require 2.81 fold and 2.71 fold increases in GPU memory than the ADS_UNet does for training. The high demand for GPU memory in the HyLT and MedFormer is not surprising, as the attention blocks introduce extra intermediate feature maps that should be kept in the GPU memory for back-propagation.

The amount of computation (FLOPs) and GPU memory requirement are the main constraints on training speed. Among all compared methods, ADS_UNet shows a clear advantage in training speed, because the lower GPU memory requirement of ADS_UNet allows us to use a larger batch size for faster training. The training speed of nnU-Net is close to ADS_UNet, for the same reason. The transformer-based models (MedFormer and HyLT) are the slowest ones since they have the highest GPU memory demand and relatively high computation cost.

3.4 Ablation Studies

3.4.1 Down-sampling masks vs. up-sampling feature maps

We build UNet and UNet as UNet's counterparts to demonstrate the advantage of using down-sampled masks for deep supervision. In UNet , feature maps of the UNet are bilinearly interpolated to fit the size of the original mask, while in UNet , the masks are down-sampled to fit the size of feature maps. As shown in Table 3.3, UNet with average pooled masks outperforms UNet by 1.21, 2.30 and 0.05 mIoU on the CRAG, BCSS, and GlaS datasets. This is achieved with only 0.06% more parameters, 1.26% more GPU memory consumption and 0.08% more FLOPs. This small increase comes from a 1 1 convolution layer appended to supervised blocks. We attribute this performance gain

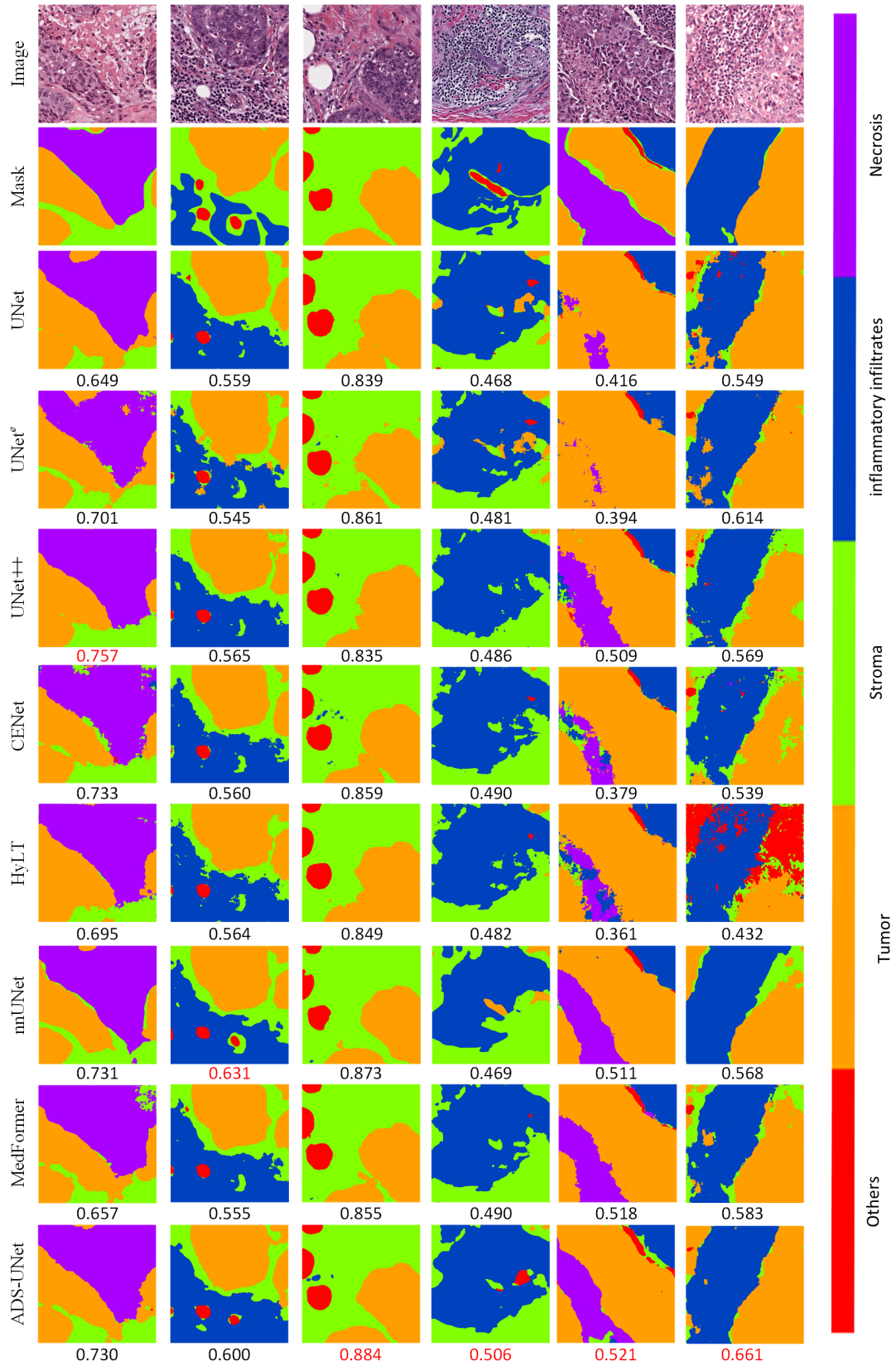


FIGURE 3.4: Visual comparison of segmentation maps. The mIoU score of each prediction is reported below the prediction.

TABLE 3.3: Comparison between the original UNet (without deep supervision) and UNet /UNet with deep supervision using up-sampled feature maps/average pooled masks.

Net	PRM	FLOPs	GPU	CRAG	BCSS	MoNuSeg	GlaS
UNet	**31.04**	**218.9**	**5.54**	84.38	71.22	86.07	91.49
				0.95	1.43	0.58	0.95
UNet	31.17	260.27	14.03	84.62	72.75	**87.18**	**92.04**
				1.33	0.71	0.34	0.90
UNet	31.06	219.08	5.61	**85.59**	**73.52**	85.41	91.54
				1.24	0.59	0.28	0.80

to back-propagation through deep layers enforcing shallow layers to learn discriminative features. UNet yields 1.77 and 0.50 higher mIoU than UNet on MoNuSeg and GlaS dataset, but with the higher computation cost. The 18.80% more computation of the UNet (compared with UNet) originates from bilinear interpolation operations when up-sampling feature maps. The GPU memory required in the training process of UNet is 2.50 times that of UNet . The reason is that during back-propagation the output of all layers is cached during forward propagation, and the size of the feature map of the supervision layer in UNet is 4 to 256 times the size of the corresponding one in UNet . Therefore, beyond a small performance improvement, UNet saves more than 1.50 GPU consumption thus enabling us to use a larger batch size and save training time. However, it is worth noting that UNet even performs worse than UNet on the MoNuSeg dataset. We argue that this is because down-sampling masks may eliminate small-sized nuclei, thus offering incorrect labelling information during training and finally resulting in inferior performance. We further validate this claim in the following sections.

3.4.2 Tracing the origin of the performance gain of ADS_UNet.

To gain insight into the reason why ADS_UNet demonstrates superior performance on segmentation, we construct eight models and evaluate them on the CRAG and MoNuSeg datasets, with each of them being a combination of deep supervision, scSE feature re-calibration blocks and sample re-weighting. The configuration of models, the performance of each constituent UNet^d and their ensemble performance are summarised in Table 3.4. To see whether the weighted average voting of base learners is better than simple average voting or not, the results of different ensemble strategies are also reported. As the results on CRAG and MoNuSeg datasets are quite different, the interpretation of Table 3.4 is organised per dataset.

Ablation on CRAG dataset. As seen in the top part of Table 3.4, when compared

with M_0 (none of three components is used), M_1, M_2 and M_4 demonstrate the effectiveness of incorporating deep supervision, scSE feature re-calibration and sample re-weighting into the training of each constituent UNet^d, respectively. Moreover, all constitute UNet^{d}s of M_1 (with scSE) surpass the counterparts of M_0, M_2 and M_4. This supports the claim we made in section 3.2.3, namely, features from the encoder block of UNet$^{d\ \ 1}$ should be re-calibrated before concatenating with features from the decoder block of UNet^d.

In M_3, M_5 and M_6, we either remove deep supervision or sample re-weighting or the scSE block from the ADS_UNet, respectively, to show the importance of each component in the composition of the ADS_UNet. As seen in Table 3.4, removing any one of them would lead to lower segmentation performance. The comparison between each pair of (M_i, $M_{i\ \ 4}$), $i \quad 0,1,2,3$ demonstrates that removing deep supervision is harmful to the performance. Further analysis is provided in section 3.5.2 to reveal the reason why introducing explicit deep supervision leads to better performance.

The comparison between each pair of (M_i, $M_{i\ \ 1}$), $i \quad 0,2,4,6$ demonstrates that truncating the gradient flow between encoder blocks of UNet^d and decoder blocks of UNet$^{d\ \ 1}$ is detrimental to the final segmentation performance. By introducing feature re-calibration in skip-connections, features learnt in encoder blocks are re-weighted to adapt to the ones of decoder blocks, thereby leading to better performance.

In terms of sample re-weighting, the ensemble (ens( )) of ADS_UNet surpasses the one of M_5 by 0.11 points. We attribute this to sample weight updating, which allows UNet^d to pay more attention to images which are hard to be segmented by UNet$^{d\ \ 1}$. The benefit of sample re-weighting is also reflected in comparisons of M_1 vs. M_3 (0.1).

When comparing ensemble strategies, we find weighted voting improves segmentation performance while average voting negatively affects the performance when compared with UNet^4. This supports the view that integrating multiple models by weighting each as per its segmenting ability improves the overall performance of the ensemble. In section 3.5.4, we further investigate how the ensemble strategy works, by evaluating the diversity of features. Figure 3.5 visualises the mIoU and assigned weighting factor ($_i$) of each UNet^i. As shown in the figure, a UNet^d achieving a higher mIoU score is assigned a larger ^{i} value, meaning that it gives more contribution when generating the final prediction.

Ablation on MoNuSeg dataset. As seen in the bottom part of Table 3.4, the effectiveness of the scSE re-calibration (M_i *v.s.* $M_{i\ \ 1}$, $i \quad 0,2,4,6$), sample re-weighting (M_i *v.s.* $M_{i\ \ 2}$, $i \quad 0,1,4,5$) and weighted ensemble strategy (*ens v.s.ens*) is clearly supported by experiments and is in line with the claim we made on the CRAG dataset. When trained on the MoNuSeg dataset, however, we observe that introducing deep supervision adversely affects the performance of each UNet^d and therefore the final

TABLE 3.4: Ablation study on CRAG and MoNuSeg datasets. Performance measured by mIoU (highest score per column is highlighted in bold). "DS" denotes deep supervision, "scSE" denotes spatial and channel squeeze & excitation used in skip-connections. "ReW" denote training sample re-weighting. "ens"/"ens ( )" denote that segmentation results from all branches are collected and then averaged, or summed by weights.

	Net	DS	scSE	ReW	UNet^1	UNet^2	UNet^3	UNet^4	ens	ens( )
CRAG	M_0	✗	✗	✗	66.26 1.13	77.35 1.30	83.11 1.29	85.11 1.23	84.10 1.22	85.62 1.10
	M_1	✗	✓	✗	69.88 0.94	80.74 0.81	85.18 0.79	**86.752** 0.79	86.05 0.74	86.79 0.79
	M_2	✗	✗	✓	66.59 1.16	77.99 1.05	83.30 0.92	85.20 0.99	84.20 0.94	85.71 0.94
	M_3	✗	✓	✓	70.49 1.13	**80.89** 0.73	85.36 0.80	86.64 0.76	86.29 0.81	86.89 0.78
	M_4	✓	✗	✗	66.74 1.17	78.16 1.20	83.64 1.12	85.73 1.20	84.67 1.13	86.01 1.15
	M_5	✓	✓	✗	**70.75** 1.21	81.30 0.78	85.11 0.88	86.61 0.92	86.27 0.71	86.81 0.82
	M_6	✓	✗	✓	66.26 1.13	77.52 1.42	83.48 1.32	85.30 1.32	84.23 1.15	85.91 1.21
	ADS_UNet M_7	✓	✓	✓	70.31 1.06	80.77 0.84	**85.36** 0.90	86.60 0.84	**86.32** 0.79	**86.92** 0.88
MoNuSeg	M_0	✗	✗	✗	78.31 0.28	82.75 0.29	90.45 0.47	90.75 0.39	90.23 0.39	90.87 0.39
	M_1	✗	✓	✗	79.73 0.41	83.57 0.39	90.83 0.38	91.06 0.51	90.64 0.42	91.23 0.42
	M_2	✗	✗	✓	78.34 0.30	82.68 0.40	90.50 0.34	90.72 0.38	90.27 0.33	90.91 0.33
	M_3	✗	✓	✓	**79.76** 0.30	**83.82** 0.58	**90.99** 0.59	**91.12** 0.44	**90.78** 0.49	**91.36** 0.49
	M_4	✓	✗	✗	79.01 0.27	81.92 0.35	86.91 0.43	86.39 0.62	85.93 0.42	86.85 0.44
	M_5	✓	✓	✗	77.85 0.32	81.29 0.34	87.48 0.46	87.27 0.62	86.30 87.61	87.61 0.58
	M_6	✓	✗	✓	79.02 0.25	81.93 0.36	86.88 0.43	86.83 0.51	86.15 0.30	87.23 0.40
	ADS_UNet M_7	✓	✓	✓	77.84 0.34	81.30 0.33	87.41 0.43	87.49 0.62	86.44 0.43	87.78 0.41

ensemble as well, by comparing $(M_i, M_{i\ \ 4})$, i $0, 1, 2, 3$). Looking back to the image patches and masks we presented in Figure 3.3, the size of the nucleus is drastically smaller than that of the glands. We suspect that the different experimental results related to deep supervision originate from the down-sampling of masks and the characteristic of the dataset itself, i.e., the size of objects to be segmented, as down-sampling masks may introduce incorrect labelling information thus misleading the training of

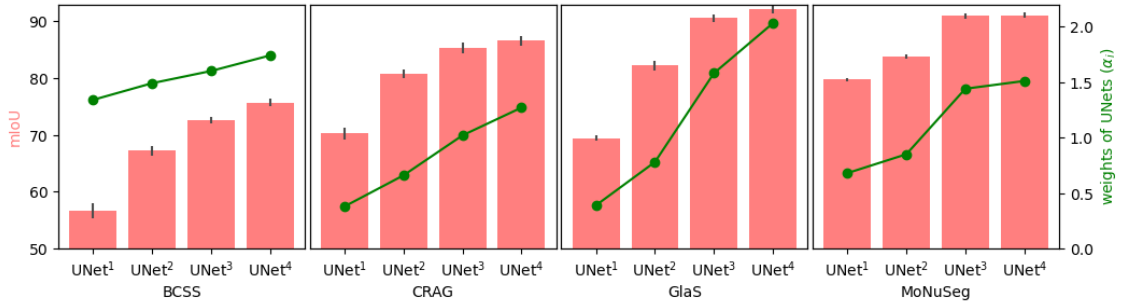


FIGURE 3.5: Visualising the performance and ^{d} weight of base UNet^{d}'. The bar chart associated with the left y-axis represents the segmentation performance of each UNet^{d}; the line chart associated with the right y-axis denotes the corresponding weighting factor (^{d}) of UNet^{d}.

hidden layers. In section 3.5.1, we quantify the incorrect labelling information to reveal the cause behind the results.

Moreover, it is worth mentioning that the segmentation performance of base UNets on the MoNuSeg dataset is almost saturated at UNet^3, and the performance gain from UNet^3 to UNet^4 is somehow negligible. Or sometimes UNet^3 is even better than UNet^4 (e.g., in M_4, M_5, M_6). This observation supports the explanation we made on why the nnU-Net with a depth of 6 achieves the worst performance on the nucleus segmentation task. i.e., when the receptive field is already large enough to completely cover the object, increasing the network depth will not enhance the feature learnt, but compressing features leads to information loss.

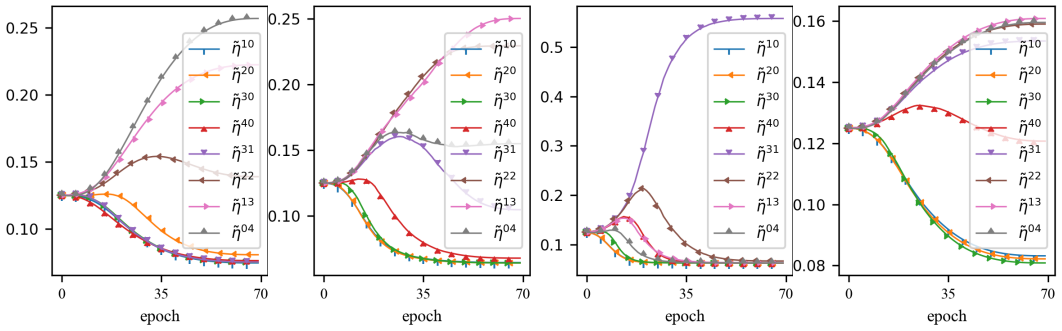
3.5 Analysis

3.5.1 Reducing incorrect labelling information by adjusting layer weights.

It is true that down-sampling the ground-truth mask eliminates small objects and leads to incorrect labels for pixels located on the class boundaries. We quantify the ratio of incorrect labels of down-sampled masks and present the statistics in Table 3.5. The code

TABLE 3.5: The proportion of incorrect labels in different scaled masks. The ($X^{i,j}$) under the down-scale factor indicates which layers the mask down-sampled by this down-scale factor is used to supervise.

Data \	None $X^{0,0}$, $X^{0,4}$	2 $X^{1,0}$, $X^{1,3}$	4 $X^{2,0}$, $X^{2,2}$	8 $X^{3,0}$, $X^{3,1}$	16 $X^{4,0}$
CRAG	0	1.01	2.96	6.45	12.49
GlaS	0	1.26	3.58	7.80	15.50
BCSS	0	1.45	4.10	9.03	17.94
MoNuSeg	0	5.24	15.25	32.53	55.26



(A) $\tilde{}^{i,j}$ values (MoNuSeg). (B) $\tilde{}^{i,j}$ values (CRAG). (C) $\tilde{}^{i,j}$ values (BCSS). (D) $\tilde{}^{i,j}$ values (GlaS).

FIGURE 3.6: (a)-(c) figures show how $\tilde{}^{i,j}_{d}$ changes when the UNet is trained on the MoNuSeg, CRAG and BCSS datasets. The changing of $\tilde{}_{i,j}$ varies from dataset to dataset. (e) The training losses of supervision layers (trained on the BCSS dataset).

for calculating the incorrect label ratio is provided in Appendix A. It can be observed that the proportion of incorrect labels rises as the down-scaling factor becomes larger. Incorrect labels in the 16 down-scaled mask in the CRAG, GlaS, and BCSS datasets account for 12.49%, 15.50%, and 17.94% of the total labels, respectively. This figure soars up to 55.26% in the MoNuSeg dataset. However, it is noteworthy that when these reduced masks are used to supervise the training of layers, there is a trainable weight $\tilde{}^{i,j}_{d}$ (defined in equation (3.4)) that dynamically adjusts the strength of each layer being supervised. Figure 3.6a-3.6c shows how the network adjusts $\tilde{}_{i,j}$ during training to assign weightings to layers and scales that contribute most to the segmentation task. As seen, at the end of the training, the largest $\tilde{}^{i,j}$ values of the MoNuSeg, CRAG, BCSS and GlaS datasets come from $\tilde{}^{0,4}$, $\tilde{}^{1,3}$, $\tilde{}^{3,1}$, and $\tilde{}^{1,3}$ respectively. That means the UNet benefits most from the original mask and the mask down-scaled by a factor of 2, 8, 2 when trained on the MoNuSeg, CRAG, BCSS and GlaS datasets. The 2 and 8 down-scaled masks carry 1.01%, 9.03% and 1.26% incorrect label information, respectively. Therefore, For CRAG, BCSS, and GlaS datasets, even though the down-scaled masks introduce wrong labelling information, the UNet is able to evade this wrong information to a certain extent and puts attention on the informative mask by adjusting $\tilde{}_{i,j}$. Despite the (apparently significant) labelling errors introduced by down-sampling, the overall results of UNet on these 3 datasets (as shown in Table 3.3) are not adversely affected. However, for the MoNuSeg dataset, due to the characteristics of the dataset itself, that is, the size of the nucleus to be segmented (and its corresponding mask) is in small size, the down-sampled mask will introduce wrong label information to a large extent, or even completely erase the nucleus. Providing overmuch misleading information in hidden layers can not be alleviated by adjusting $\tilde{}^{i,j}$, thus adversely affecting the learning process of the network, leading to inferior segmentation. We regard this as the reason that ADS_UNet trained on the MoNuSeg dataset performs worse than its counterparts wherein the deep supervision is not included (see Table 3.4 for reference).

3.5.2 Deep Supervision in UNet and ADS_UNet

3.5.2.1 Different layers contribute differently at different time stamps.

In UNet^e and UNet++, all losses have the same weight in the back-propagation process, while in UNet and ADS_UNet, $\tilde{}_{i,j}$ is trainable. The purpose of this design is to check whether all layers in the summand of the training loss in equation (3.5) contribute equally. Taking Figure 3.6b as an example, the importance of decoder nodes $X^{1,3}$ and $X^{2,2}$ is ranked in the top two. This means features learned by these 2 layers contribute more than others, with changes in their importance throughout the training process. From the perspective of back-propagation, this means that parameters of layers which have larger $_{i,j}$ values, will have relatively large changes when they are updated using gradient descent. This fact, therefore, indicates the importance of the features derived at that length scale to the separability of texture labels. A similar trend in the changes to $\tilde{}^{i,j}_{d}$ in the iterative training process of ADS_UNet is also observed in Figure 3.7e- 3.7h. Figure 3.6c and Figure 3.7e- 3.7h not only show us how the parameters of different layers change during the training process, but also indicate that: 1) the importance of parameter varies from layer to layer; 2) the significance of parameters also vary throughout the training process. This is the effect of normalisation of the weights $\tilde{}^{i,j}_{d}$, which introduces competition between the layers. And also, 3) the competition between the layers will continue until equilibrium is reached.

3.5.2.2 Preventing layer weights from vanishing leads to higher segmentation performance.

In equation (3.4), we redefine $^{i,j}_{d}$ as $\tilde{}^{i,j}_{d}$ to enforce all blocks to learn features that are directly discriminative for classifying textures. We then sum the probability maps produced by these blocks based on their importance factors $\tilde{}^{i,j}_{d}$ to generate the segmentation map of UNet^d (defined in equation (3.6)). To verify if this constraint range and the weighted combination yield better performance or not, we train ADS_UNet in 3 modes:

1) $\boldsymbol{\eta}_d$: with its element $^{i,j}_{d}$ being trained without range constraint. After UNet^d is trained, the output of the layer which has the largest $^{d}_{i,j}$ value is selected to generate the final segmentation map. i.e., let i , j arg max $_{i,j}$ $^{i,j}_{d}$, the final probability map is obtained by $\hat{y}_d$ $\hat{y}^{i}$ ^{,j} , with $\hat{y}^{i,j}$ defined in equation (3.1). Then $\hat{y}_d$ is used to compare with the ground truth to calculate the ^{d} (the weight of UNet^d).

2) $\tilde{\boldsymbol{\eta}}_d$: $\tilde{}^{i,j}_{d}$ is bounded in $\frac{1}{2}\frac{}{d}\frac{}{1}\frac{d}{2}\frac{2}{d}\frac{}{1}$, according to equation (3.4). The final segmentation map generation and ^{d} calculation criteria are the same as 1).

TABLE 3.6: mIoU score of ADS_UNet trained in 3 modes. Each UNet^d is trained for 70 epochs.

	Constraint	UNet^1		UNet^2		UNet^3		UNet^4		ens( )	
BCSS	η^d	40.09	0.97	54.15	1.11	63.88	1.14	66.09	1.25	68.34	1.12
	$\tilde{\eta}^d$	**56.87**	1.39	67.15	1.23	72.67	0.67	73.76	0.80	74.87	0.71
	$\tilde{\eta}^d$(sum)	56.74	1.34	**67.23**	0.90	**72.68**	0.57	**74.99**	0.56	**75.73**	0.60
CRAG	η^d	61.34	1.08	73.16	1.30	80.89	1.16	82.37	2.13	83.57	1.22
	$\tilde{\eta}^d$	**70.39**	1.06	80.20	.074	85.13	0.68	**86.62**	0.79	86.86	0.74
	$\tilde{\eta}^d$(sum)	70.31	1.06	**80.77**	0.84	**85.36**	0.90	86.60	0.84	**86.92**	0.88
GlaS	η^d	62.06	0.45	73.83	0.28	84.21	0.47	87.90	0.48	88.48	0.53
	$\tilde{\eta}^d$	**69.86**	0.82	**83.03**	**0.93**	90.36	0.68	91.97	0.85	92.02	0.95
	$\tilde{\eta}^d$(sum)	69.43	0.51	82.23	0.79	**90.60**	0.69	**91.98**	0.64	**92.65**	0.71

3) $\tilde{\eta}_d$ (sum): training criteria is the same as 2). While the segmentation map produced by model UNet^d is the weighted summation of multi-scale prediction (using equation (3.6)), which is then used to calculate the ^{d}.

Note that we do not run experiments on the MoNuSeg dataset, as the previous ablation study (Table 3.4 and section 3.5.1) indicates that supervising the learning of hidden layers leads to worse performance, due to the small size of the nucleus. The results of training ADS_UNet in 3 different modes are reported in Table 3.6, where all of ADS_UNets with bounded $\tilde{\eta}_d$ surpass the unbounded ones. After combining the probability maps produced by supervision layers based on the layer importance factors $^{\sim i,j}_{d}$, the mIoU score is further improved by 0.86, 0.06, 0.63 points on BCSS, CRAG, GlaS datasets. To explain the results of Table 3.6, the loss, $^{i,j}_{d}$ and $^{\sim i,j}_{d}$ of the ADS_UNet (trained in mode 1 and mode 3) are tracked and visualised in Figure 3.7. As observed in Figure 3.7i-3.7p, when there is no range constraint on $^{i,j}_{d}$, only one specific layer's loss dominates the learning process and the loss of other layers is almost negligible ($^{i,j}_{d}$ close to 0), after training for a few epochs. But the loss increases ($\mathcal{L}^{3,0}$ in Figure 3.7k and $\mathcal{L}^{4,0}$ in Figure 3.7l) indicates there is reduced discriminability at the intermediate layers ($X^{3,0}$, $X^{4,0}$) still. However, this phenomenon is eliminated after the range constraint is imposed, to suppress the weight of the dominant layer and to enable those of the others to grow, as shown in Figure 3.7a- 3.7h. That means, by retaining the information from previous layers, the range of features that are being learned is increased, therefore leading to better performance. Note that $\mathcal{L}^{3,0}$ in Figure 3.7c and $\mathcal{L}^{4,0}$ in Figure 3.7d keep decreasing, and differs from that of Figure 3.7k and Figure 3.7l.

3.5.3 Feature Similarity of Hidden Layers

Since deep supervision provides features of intermediate blocks with a direct contribution to the overall loss, the similarity of features learned by these blocks will be higher

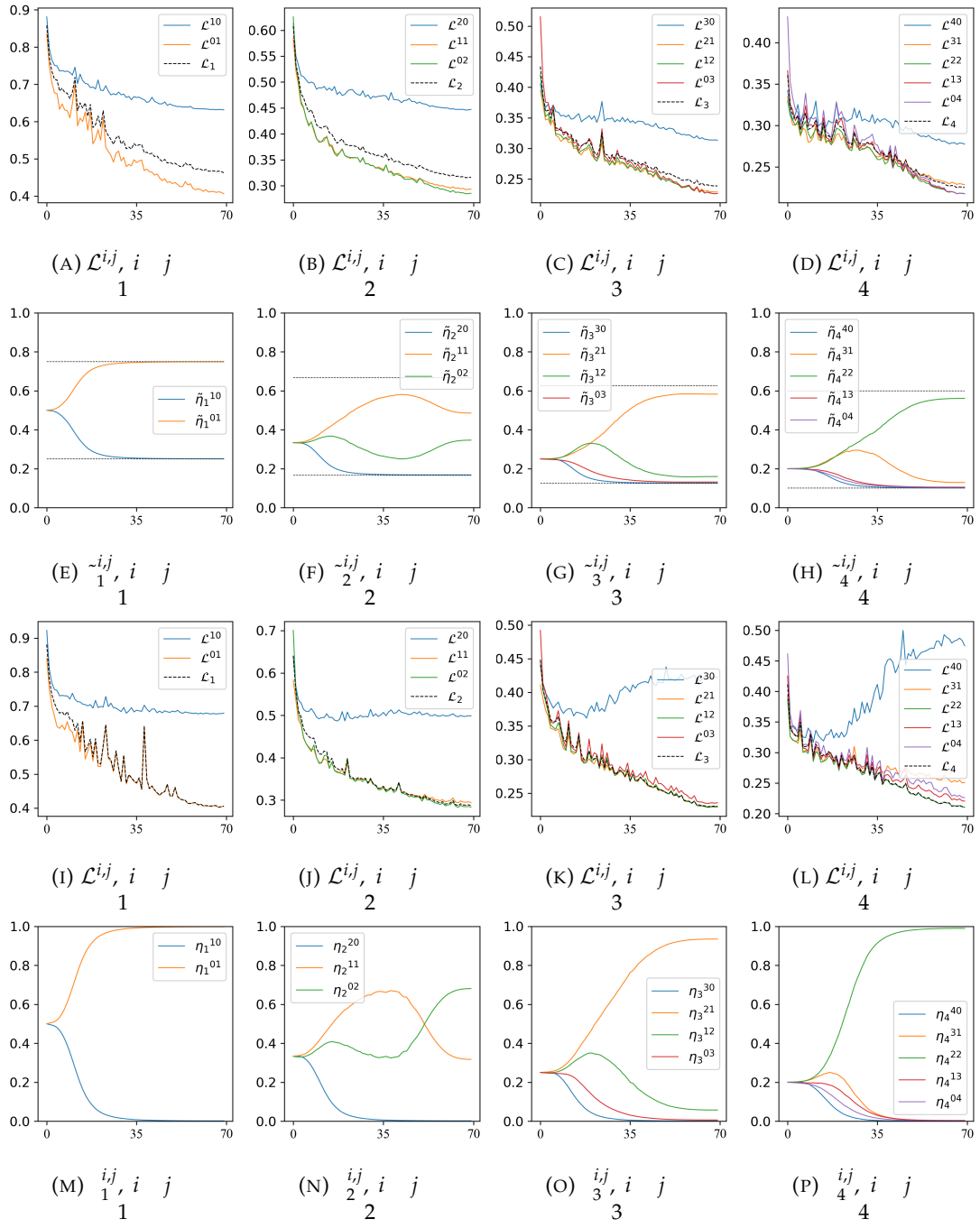
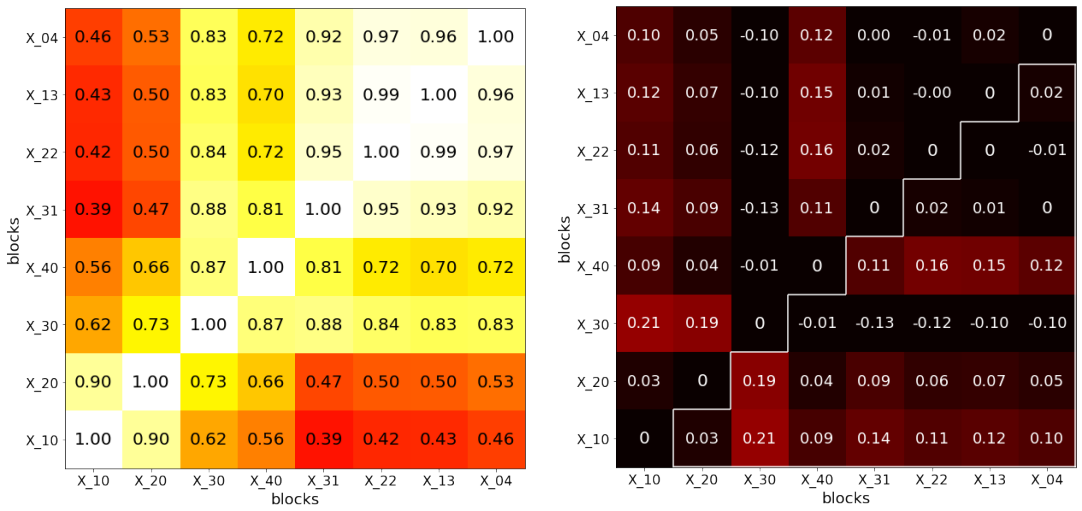


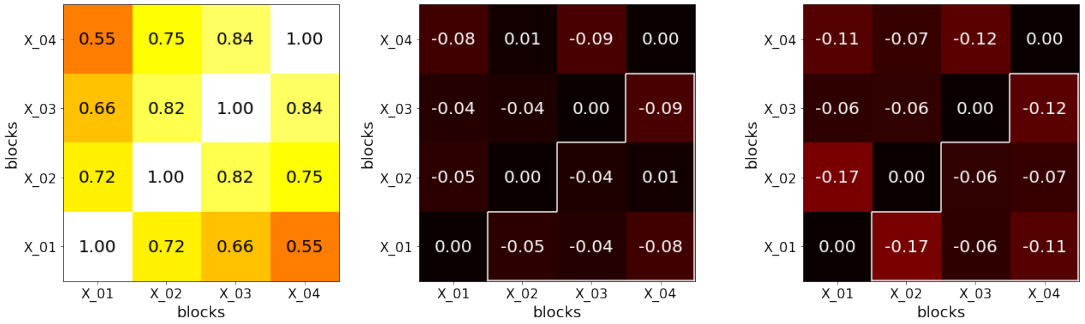
FIGURE 3.7: (a)-(d) Cross-entropy losses of supervision layers during the UNet^d training process (Equation (3.4) is imposed to constraint the range of $\tilde{\eta}_d^{i,j}$). $\mathcal{L}_d$ is calculated from equation (3.5). (e)-(h) The corresponding weights of supervision layers. $\tilde{\eta}_{ij}^{d}$ reflects the importance of node $X^{i,j}$ while computing the overall loss. (i)-(p) The loss and the $\eta_{i,j}^{d}$ values of supervision layers of UNet^d, in which $\eta_{i,j}^{d}$ is trained without constraints. $\mathcal{L}_d$ shown in (i)-(l) is calculated from equation (3.3). For all plots, the x-axis indexes the training epoch. These plots are based on the BCSS dataset.

than those of the original UNet. Centered Kernel Alignment (CKA) [55] has been developed as a tool for comparing feature representations of neural networks. Here we use



(A) UNet (B) UNet -UNet

FIGURE 3.8: (a) Feature similarity of layers for UNet. (b) The difference in feature similarity of layers between UNet and UNet. In (a), each entry shows the CKA similarity between the two layers. In (b), we calculate the feature similarity matrix for UNet , then take the difference between UNet and UNet. These plots are based on the BCSS dataset.



(A) ADS_UNet (B) ADS_UNet UNet^e (C) ADS_UNet UNet++

FIGURE 3.9: (a) Feature similarity of the output layers of ADS_UNet. (b) and (c) We calculate the feature similarity matrix for UNet^e and UNet++, then take the difference between ADS_UNet and UNet^e, UNet++. These plots are based on BCSS dataset.

CKA to characterize the similarity of feature representations learned by different convolutional blocks in UNet . As shown in Figure 3.8b, the similarity of features extracted by blocks in UNet is mostly higher than in their counterparts in UNet (although 6 of similarity entries in UNet have lower values than that of UNet), which is consistent with our expectation (the 20 positive values add up to 1.89 vs. the 6 negative values add up to -0.47).

3.5.4 Feature Diversity of Output Layers

Ensemble-based learning methods, such as AdaBoost, rely on the independence of features exploited by classifiers in its ensemble [31]. If base learners produce independent

outputs, then the segmentation accuracy of the ensemble can be enhanced by majority weighting. Figure 3.9a characterize the feature similarity of output layers of ADS_UNet. Figure 3.9b and 3.9c shows that features learned by the output layers of ADS_UNet are less similar than those in UNet^e (the values add up to -0.29) and UNet++ (the values add up to -0.59). Our interpretation is that this can be attributed to the stage-wise additive learning, followed by the sample weight updating rule of ADS_UNet, and may explain why ADS_UNet outperforms UNet^e and UNet++.

3.6 Summary

In this chapter, we propose a novel stage-wise additive training algorithm, ADS_UNet, that incorporates the AdaBoost algorithm and greedy layer-wise training strategy into the iterative learning progress of an ensemble model. The proposed method has the following advantages: 1) The stage-wise training strategy with re-weighted training samples empowers base learners to learn discriminative and diverse feature representations. These are eventually combined in a performance-weighted manner to produce the final prediction, leading to higher accuracy than those achieved by other UNet-like architectures. 2) In the configuration of base learners, intermediate layers are supervised directly to learn discriminative features, without the need for learning extra upsampling blocks. This, therefore, diminishes memory consumption and computational burden. 3) By introducing layer competition, we observe that the importance of feature maps produced by layers varies from epoch to epoch at the training stage, and different layers contribute differently in a manner that is learnable. 4) ADS_UNet is more computationally efficient (fewer requirements on GPU memory and training time) than UNet^e, UNet++, CENet and transformer-based UNet variants, due to its cascade training regimen.

However, the ADS_UNet has the following limitation that we would like to address in future work:

- Currently, the sample re-weighting training criteria restricts the ADS_UNet to only update the weights of samples at a relatively coarse granularity. In future work, more fine-grained re-weighting criteria will be explored to guide successive base learners to pay more attention to regions/pixels that are difficult to distinguish.
- We currently manually decide whether deep supervision is applied or not, based on our visual inspection of the datasets' characteristics. It would be interesting to explore some decision-making methods that can automatically make choices about whether deep supervision should be included or not.

- We are also interested in absorbing and extending the systematic heuristic rules of nnU-Net into our ADS_UNet architecture or exploring neural architecture searching (NAS) techniques that optimise the configurations (depth of the architecture, the size of filters, etc.) of the network automatically. We believe that making the framework completely automatic would drive the application of deep learning in medical imaging further.

Chapter 4

Scale Equivariance for Robust Segmentation

4.1 Motivation and Contribution

Pathologists diagnosing biopsy samples view histopathology slices at different magnifications by controlling the microscope's objective revolver. Neural network based decision support for digital pathology takes as input digital images scanned from glass slides. Specimen slides scanned at different medical institutions may use different objective magnifications to digitalise specimen slides, resulting in whole slide images (WSI) being at different scales. For example, images provided by the CRAG dataset [3] are in 20 magnification; For the DigestPath-2019 dataset [66], images are in 40 magnification. Such a large difference in imaging magnification makes digitised WSIs look significantly different in appearance (see Figure 4.1 for an example). Models such as Convolutional neural networks (CNNs) trained on images at a specific scale generally can not generalise to other scales, which greatly restricts the applicability of computer-aided diagnosis models.

It is true that in general the magnification used to digitalise biopsy specimens is known and WSIs are usually stored in a pyramid structure (as shown in chapter 1, Figure 1.1), this makes it possible for the end user to re-scale the WSI to fit it with the neural network trained on images presented at a specific scale. This will thus eliminate the generalisation issue caused by the magnification level mismatch between the training set and the test set. However, the scale/magnification generalisation problem still exists in some scenarios where the magnification level is unknown, for example, *snapshots*. In a practical clinical diagnosis procedure, most pathologists still use light microscopy to examine the tissue and snapshots of critical field-of-views are captured using mounted cameras for histology examination reports or case studies. Nevertheless, such an invaluable collection of snapshots is usually archived without magnification information.

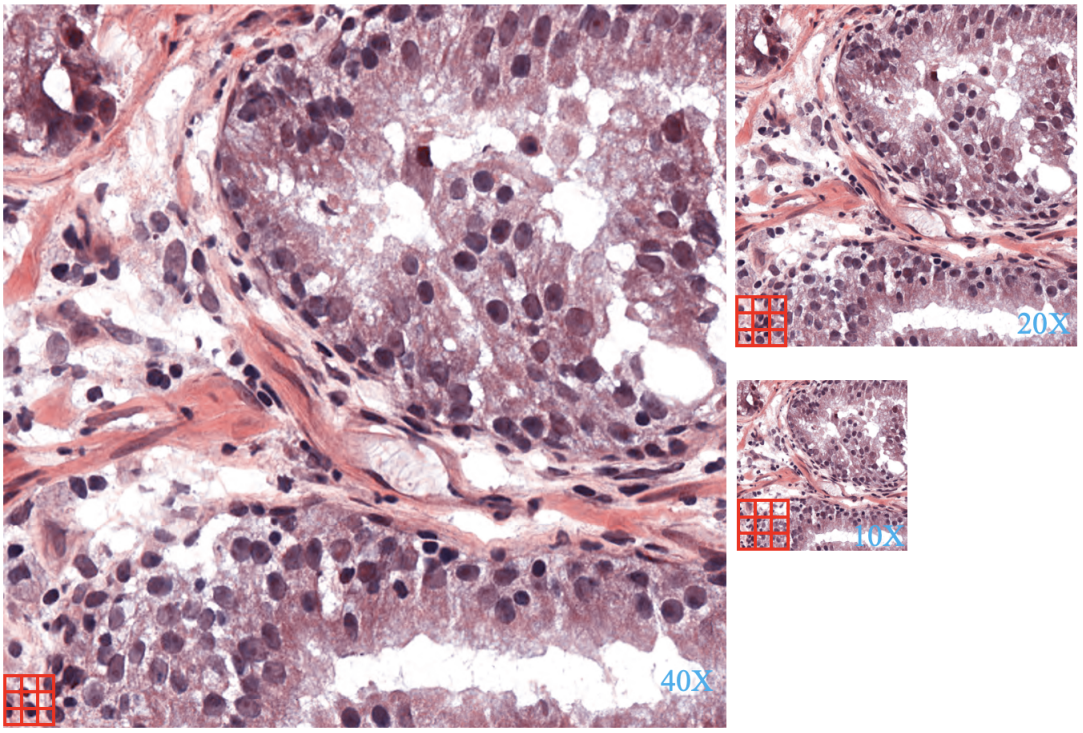


FIGURE 4.1: Hematoxylin and Eosin (H&E) stained histopathology images were presented in 40 , 20 and 10 magnifications (clockwise). The 3 3 filter presented at the bottom left corner of images demonstrates that a filter of fixed size covers the different extent of pixels for images in different scales.

Such loss of magnification information hinders the full use of microscopic snapshots. It has been witnessed that there are very rare studies that use snapshots for training neural networks for downstream tasks such as classification, detection, segmentation, etc., even though they are very representative and carry valuable features and disease knowledge. To remove the barrier brought about by the unknown magnification, previous work trains neural networks to recognise or predict the magnification of given snapshots by either treating it as a classification problem [6, 107, 124] or a regression problem [84, 125]. Once the magnification level is recognised precisely, the snapshots can be re-scaled and fed into models for further analysis. We admit that the work of magnification recognition could advance the full use of snapshots, however, building models that are less sensitive to magnification variance is more appealing.

CNNs have dominated the computer vision field since the proposal of the AlexNet [57]. The most widely adopted strategy to cope with scale variation in unseen data is introducing scale augmentation during training CNNs, where training samples are randomly scaled before being fed into the network. Other attempts such as scale selection [37] and scale fusion [53] also help to circumvent scale changes. However, these methods lack explicit mechanisms to model scale information. Some works such as [50, 76, 120] achieve scale equivariance by resizing the input or filter, but these methods

are computationally expensive since they rely on tensor resizing and image interpolation. Other ways of generating filters of different sizes include [8, 96, 130], parameterising filters by a trainable linear combination of a family of predefined, fixed multi-scale basis functions (B-Splines, Hermite, Fourier). Such methods, however, require that both the scale of basis functions and the size of filters should be fixed, once the network has been initialised. The work presented in Pintea et al. [87] shows that hard-coding the scale hyper-parameters in the network can be restrictive while learning the scale parameter is especially beneficial when dealing with inputs at multiple resolutions.

In this chapter, we introduce the Scale-Equivariant UNet (SEUNet), which demonstrates superior generalisation performance on image datasets at different scales when compared with the conventional CNN model and other scale-equivariant models. The main characteristics of our work are as follows:

- We parameterise convolutional filters with learnable Gaussian derivative filters, instead of using a set of pre-calculated, fixed filter basis.
- We impose range constraints on learnable scale parameters to ensure coverage of multiple scales, while allowing them to be tuned within disjoint intervals. This frees up model capacity to find an optimal set of scale parameters that adapt to training samples by back-propagation.
- Since there is no dependency between feature maps produced by filters at different scales, the model can be viewed as a union of multiple independent sub-models, each of which can be used separately to save GPU memory consumption and accelerate inference.

4.2 Methodology

Our work extends Lindeberg [70] from image classification to image segmentation while also allowing the of each layer to be learnable similar to [87]. Before moving into the construction of scale equivariant convolution, we give a brief introduction to scale transformation, scale equivariance, and steerable filters.

Scale transformation. The scaling operator T_s is defined on a function (image) f thus:

$$T_s f \quad x \qquad f \; s \; {}^{1}x \; , \; s \quad 0, \tag{4.1}$$

where s denotes a scaling factor, we refer to cases with s 1 as up-scalings and to cases with s 1 as down-scalings.

Scale-Equivariance. For a family of feature mapping operators (typically a convolutional layer), scale equivariance means that the scale transformation should commute

with the feature mapping operation according to

$$T_s f \quad T_s \quad f \quad , \tag{4.2}$$

where denotes some feature map operators within the same family that operates on the image re-scaled by a factor of s.

Steerable Filters [30]. Steerable filters refer to a class of filters wherein a filter of arbitrary orientation and scale is synthesised as a linear combination of a set of "basis steerable filters".

4.2.1 Gaussian derivative filters are equivariant under scale transformation

We construct the convolutional filter by linear combining 2D Gaussian derivative basis. The 1D Gaussian filter at scale is written as $G\ x; x_0, \quad \frac{1}{2} e^{\frac{x\ x_0}{2}^2}$ which can be extended to 2D isotropic Gaussian filters as $G\ x, y; x_0, y_0, \quad G\ x; x_0, \quad G\ y; y_0, \quad .$ We will drop the centres x_0, y_0 to simplify notation. In [70, 87, 121], the authors linearly combine the 2D Gaussian derivatives,

$$G^{i,j}\ x, y; \quad \frac{^{i\ j}G\ x, y;}{x^i\ y^j} \quad \frac{^{i}G\ x;}{x^i} \frac{^{j}G\ y;}{y^j}, \ i, j \quad 0, \ i \quad j \quad N \tag{4.3}$$

to construct filters. N refer to the highest order of derivative used in equation (4.3).

Considering two images f and f that are related by a scaling transformation $f \quad x\ , y$ $f\ x, y$ for $x \quad sx, y \quad sy$, where $s \quad 0$ is a scaling factor. The scale-space representation of f and f are denoted as $L\ x, y; \quad G^{0,0}\ , \ ; \quad f\ , \quad x, y$ and $L\ \ x\ , y\ ;$ $G^{0,0}\ , \ ; \quad f\ , \quad x\ , y\ ,$ respectively. denotes 2D convolution. If the scale parameters and of images f and f are related according to $s\ ,$ the authors of [69] have proved that Gaussian scale-space representations are equal at matching image points and scales:

$$L\ x, y; \quad L\ \ x\ , y\ ; \quad . \tag{4.4}$$

Regarding Gaussian derivatives, the general result in equation (4) of [69] suggest that the spatial derivatives of a Gaussian filter are also related under scaling transformation,

$$\begin{aligned} L_{i,j}\ x, y; \quad &: \quad G^{i,j}\ , \ ; \quad f\ , \quad x, y\ , \\ L_{i,j}\ x\ , y\ ; \quad &: \quad G^{i,j}\ , \ ; \quad f\ , \quad x\ , y\ \ , \\ Li, j\ x, y; \quad &L_{i,j}\ x\ , y\ ; \quad , \end{aligned} \tag{4.5}$$

where $i \quad j \quad N, \ i, j \quad 0$. Therefore, equations (4.4) and (4.5) show that the elements of a filter basis are scale-equivariant, and so is the filter constructed from linear combinations of the basis.

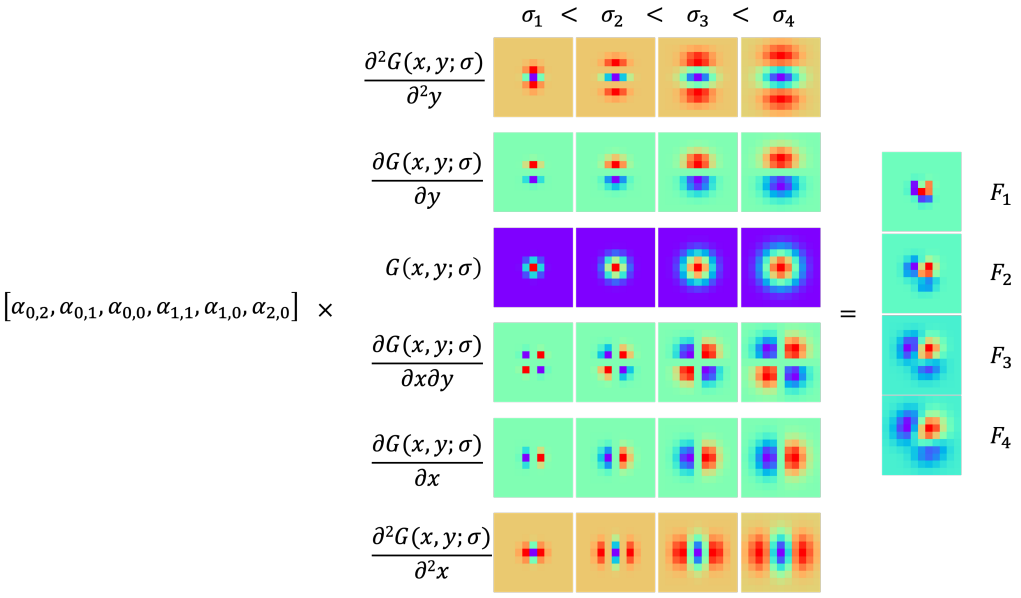


FIGURE 4.2: The graphical illustration of filter construction (equation 4.6). The filter constructed here is based on the order N 2. As shown on the right, by setting different values, the synthesised filters are in the same shape but vary on scales. The channel index c and layer index l are ignored For simplicity.

.

4.2.2 Parameterising convolutional filters, layer by layer

Filter construction. In conventional CNNs, a bank of filters F^l of size $C_l, C_{l\ 1}, h, w$ is used to map an input image f^0 or feature map $f^{l\ 1}$ $C_{l\ 1}$ H W into f^l C_l H W by convolution. Here l 1 is the layer index, H, W and h, w denote the size of the feature map and the size of the filter, respectively. Zero padding is applied so that the size of the feature map output remains the same as the input. We compute scale-space feature maps by convolving with groups of filters at different scales, with each convolutional filter a linear combination of Gaussian derivative filters:

$$F_k^l\ c_l, x, y,\ {}^l_k; c_{l\ 1} \quad {}^{i\ j\ N}_{i,j\ 0}\ {}^l_{i,j,c_l,c_{l\ 1}} G^{i,j}\ x, y;\ {}^l_k\ , \tag{4.6}$$

where ${}^l_{i,j,c_l,c_{l\ 1}}$ are learnable, and independent of k which indexes scale settings within a layer. A graphical illustration of equation 4.6 is provided in Figure 4.2. We describe these next, first by explaining the first layer from image to features, and then by showing how features are composed in subsequent layers. In detail, the C_l channels in F^l are divided into groups denoted by F_k^l ${}^{C_l}_{\ }$ $C_{l\ 1}$ h w, k 1, . . . , . The first layer maps the input image f^0 into f^1 F^1 f^0, f^1 ${}^{C_1}_{\ }$ H W, by convolving with filters $F_k^1\ c_1, x, y,\ {}^1_k; c_0$ at position x, y , input channel c_0 , , and output channel c_1 1, , ${}^{C_1}_{\ }$. The first dimension in f^1 represents the scale axis, with scales indexed by k. Note that the subscript k of ${}^l_k$ denotes that the scale parameter

varies across groups in the same convolutional layer l, for all l, but is shared across filters in the same group. The $\frac{l}{i,j,c_l,c_{l\ 1}}$ in equation (4.6) does not have a group index k in its subscript, as we share these learnable weights between groups, in order to ensure that the convolution kernels generated in different groups are consistent in shape, and do not mix separated scale factors, thus ensuring scale equivariance. When 1, the F^l degenerates to N-Jet convolutional filter in which the is shared for the complete layer [87]..

Scale convolution in hidden layers. For layers l 2 feature maps are divided into groups $f^{l\ 1}$ $\frac{C_{l\ 1}}{}$ H W, each representing the response to a specific scale in $f_k^{l\ 1}$ $\frac{C_{l\ 1}}{}$ H W. We again use equation (4.6) to construct groups of filters F^l F_1^l, , F^l , F_k^l $\frac{C_l}{}$ $\frac{C_{l\ 1}}{}$ h w to convolve with $f_k^{l\ 1}$. After the first layer, we define the network architecture to have f_k^l F_k^l $f_k^{l\ 1}$: in subsequent layers, each group of filters acts only on a subset of scale-matched channels. denotes 2D convolution. This is in contrast to the first layer (l 1) where the filters act on the entire image to generate f_k^1 : $f^{1\ k}$ F_k^1 f^0, with all colour channels contributing to the scale-specific channels in f^1 indexed by k. We thus have the learnable coefficients and $\frac{l}{k}$ range over channel indices

$$\frac{l}{i,j,c_l,c_{l\ 1}}\ c_0 \quad , \quad , \quad , c_l \quad 1,\ldots,\frac{C_l}{} \quad \text{and} \quad \frac{l}{k}\ k \quad 1,\ldots, \quad . \tag{4.7}$$

Thus, the propagation of information captured by the composition of layer-wise convolutions does not mix information from different scales. The restriction on network connections to scale-matched layer outputs is designed to maintain equivariance to input rescaling at layer outputs under composition. Although acting F_k^l on the entire f^l can be another option, an attempt in [96] shows that introducing inter-scale interaction also introduces extra equivariance error, and leads to lower performance. We further tune the successive scale factors $\frac{l}{k}$ to track the increase in the receptive field with depth.

4.2.3 Imposing range constraints on scale parameters

The trainable parameters in equation (4.7) in the filters include the scale parameters $\frac{l}{k}$ learned during back-propagation. However, leaving them to be tuned completely freely may lead to a problem: all $\frac{l}{k}$s in the same layer may have the same value, which means constructed filters are redundant, limiting the scale diversity of filters. As our original intention is that the network can achieve scale equivariance by learning multi-scale convolutional filters, we introduce the following constraints to separate $\frac{l}{k}$ values to lie in disjoint intervals.

$$\frac{l}{k}\ x \quad \frac{a_k^l \quad b_k^l}{2}\tanh x \quad \frac{a_k^l \quad b_k^l}{2},\ a_k^l \quad b_k^l,\ b \quad 0 \tag{4.8}$$

where a_k^l and b_k^l are hyper-parameters for the upper and lower bounds for σ of filters at the l^{th} layer and the k^{th} group. x is a trainable real variable. By setting an appropriate set of a_k^l and b_k^l: multi-scale filters can be constructed as per equations (4.6). Once σ_k^l is known, the following formula used in [87] is employed to determine the size τ_k^l of filters f_k^l:

$$\tau_k^l = 2\left\lceil 2\sigma_k^l \right\rceil + 1,\ \sigma_k^l > 0 \tag{4.9}$$

This enables us to train the size of the receptive field. In the encoder path of the UNet (layer 1-8), we gradually increase σ, to increase receptive field size. This is in keeping with Lindeberg [70]. In the decoder path (layer 11-18), we gradually decrease σ. Layers 9-10 are the bottleneck layers. An ablation study with regard to the setting of σ_k^l can be found in section 4.4 which demonstrates the benefit of imposing range constraints on σ_k^l.

4.2.4 Parallelising training by simultaneous optimisation of multiple loss functions

As described in section 4.2.2, filters in each layer are divided into γ groups and each group of filters operate only on an non-overlapping subset of feature maps with no inter-scale feature interactions. Thus, features learned by these γ groups of filters can be trained in a mutually independent fashion. A penultimate convolution layer for feature fusion creates a score that is passed to a softmax function, followed by the calculation of loss function. This, however, mixes multi-scale information and destroys the scale equivariance of the features. Therefore, to train all groups of filters simultaneously while maintaining equivariance between multi-scale features, we propose to minimise a weighted combination of multiple loss functions, with each of them acting only on a single group of filters. In detail, given the ground truth y and feature map f_k^{L-1} that is produced by filters F_k^{L-1} in a L-layer network, a 1×1 convolution with softmax activation is used to map f_k^{L-1} into a probability map $\hat{y}_k$ for each of C classes for every pixel in the $H \times W$ image. The loss function used to train F_k^{L-1} is the norm cross-entropy loss:

$$l_k(y, \hat{y}_k) = -y\log(\hat{y}_k),\ y \in \{0,1\}^{C\times H\times W}, \hat{y}_k \in \mathbb{R}^{C\times H\times W},\ k = 1, \ldots, \gamma. \tag{4.10}$$

The overall loss function is defined as:

$$L = \sum_{k=1}^{\gamma} \tilde{\eta}_k l_k(y, \hat{y}_k),\ \tilde{\eta}_k = \frac{\eta_k + \frac{1}{\gamma}}{\sum_{k=1}^{\gamma}(\eta_k + \frac{1}{\gamma})},\ \text{where } 0 \le \eta_k \le 1,\ \sum_{k=1}^{\gamma}\eta_k = 1,\ \sum_{k=1}^{\gamma}\tilde{\eta}_k = 1. \tag{4.11}$$

η_k is a weighting factor that is assigned to F_k to characterise the relative importance between scales. It is quite plausible that the loss function is minimised by a dominating contribution from a specific scale indexed by k, driving all other η_k to zero, a

phenomenon called competitive exclusion. It is to maintain some contribution from features acquired at multiple scales that we introduce the additive constant $(1/\gamma)$ in $\widetilde{\eta}_k$. This constrains the trainable $\widetilde{\eta}_k$ to be in the range $[\frac{1}{2\gamma}, \frac{\gamma+1}{2\gamma}]$. In practice, we initialise η_k to $\frac{1}{\gamma}$ and use the *softmax* function to normalise η_k to guarantee $\sum_{k=1}^{\gamma} \eta_k = 1$. Figure 4.3 shows the entire structure of the model proposed here.

4.2.5 Final Prediction Generation

The SEUNet generates probability maps $\hat{y}_{k,n}$, $k = 1, \ldots, \gamma$ from learned filters with different scales/sizes for each pixel n. For each pixel n, let $\hat{y}_{k,n,c}$ be the probability of predicting class $c \in C$ by classifier indexed by scale k. Given an image with unknown scale information and these γ probability maps, we explore the following strategies to generate the final segmentation map.

Arithmetic mean ensemble. For each pixel n the final segmentation map is obtained from $\arg\max_c (1/\gamma) \sum_k \hat{y}_{k,n,c}$. We denote this strategy P_Arithm.

Prediction selection based on prediction confidence. Let $(k, n, c^*) = \arg\max_c \hat{y}_{k,n,c}$ and $(k, n, c') = \arg\max_{c \neq c^*} \hat{y}_{k,n,c}$. Then $\delta_{n,k} := (\hat{y}_{k,n,c^*} - \hat{y}_{k,n,c'})$, the difference between the largest and second-largest class probability is a measure of the predictive confidence

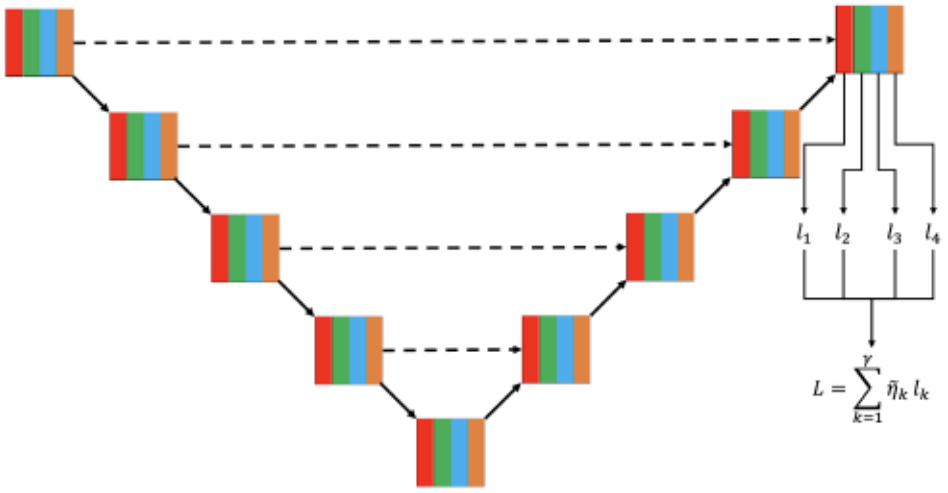


FIGURE 4.3: The architecture of the Scale-Equivariant UNet (SEUNet) model proposed here. Each square node in the graph represents a convolution block that consists of two convolutional layers. In each square node, four rectangles in different colours denote four groups of filters that are parameterised by different σ_k^ls, but share the same α. All filters with the same colour form an independent sub-network, and all sub-networks have their own prediction and loss functions $\{l_k \mid k \in \{1, \cdots, 4\}\}$. All sub-networks can be trained simultaneously in an end-to-end fashion by minimising the combined loss function.

of classifier k. We choose the most *confident* prediction (k $\arg\max_k$ $_{n,k}$, so c $\arg\max_c \hat{y}_{k\ ,n,c}$) for pixel n as its final predicted label. We denote this strategy P_Dist.

Prediction ensemble based on prediction confidence. To mitigate against the concern of an incorrect prediction made with high confidence, we propose P_Ens, a per-pixel ensemble strategy that weights multiple predictions based on their confidence. Thus multiple less confident predictions can compensate in test cases where the highest confident prediction may be incorrect. The detailed process of generating the final prediction using P_Dist or P_Ens strategies is described in algorithm 3. We first calculate the prediction confidence, the difference between the largest and the second largest probability, then the weighting factor w_k of prediction $\hat{y}_k$ is determined by the *softmax* function, giving larger weights to confident predictions.

Algorithm 3: Per-pixel Prediction Selection/Ensemble based on Probability Distance.

Input: Probability vectors $\hat{y}_1$, , $\hat{y}$, $\hat{y}_k$ ^{C}, k 1, , ; C is the number of classes. *Strategy* *P_Dist* or *P_Ens* .

D ;
for k 1, 2, ..., **do**
  p_{max} = max($\hat{y}_k$)
  p_{max_idx} = arg max $\hat{y}_k$)
  $\hat{y}_{k,p_{max\_idx}}$ 1
  p_{second_max} = max($\hat{y}_k$)
  D.append p_{max} p_{second_max}
  $\hat{y}_{k,p_{max\_idx}}$ p_{max}
end
if *Strategy==P_Dist* **then**
  k arg max D
  $\hat{y}$ $\arg\max_C \hat{y}_k$
end
if *Strategy==P_Ens* **then**
  for k 1, 2, ..., **do**
    w_k $\frac{e^{D_k}}{\sum_{k\ 1} e^{D_k}}$
  end
  $\hat{y}$ $\arg\max_C w_k \hat{y}_k$
end

Output: $\hat{y}$

4.3 Experiments and Results

4.3.1 Datasets and Evaluation Setting

Datasets. We use CRAG [3], GlaS [94] and MoNuSeg dataset [58], which are introduced in section 3.3.1, for evaluating our method. For each dataset, 5-fold cross-validation is used to measure the performance.

Evaluation regime. Since we aim to evaluate the models' generalisation capability with respect to images' scale variation, we re-scale the test set by a series of scale factors between 0.25 and 4, with a relative scale ratio of $\sqrt[4]{2}$ between adjacent testing scales. This setting is in line with [70].

4.3.2 Compared methods

We use the standard UNet architecture as a backbone and replace the conventional convolutional layers with different types of scale-equivariant convolution to generate four scale-equivariant UNet variants: SESN-UNet, DISCO-UNet, SDCF-UNet, and the proposed SEUNet. For the UNet with conventional convolutional layers, the number of filters at each depth is 64, 128, 256, 512, 1024. For a fair comparison, all of the UNet variants have the same number of scales (refers to the hyper-parameter). For the SESN, SDCF and DISCO model, we set as 4 and scale factors as $1, 2, 3, 4$, therefore the size of filters at each scale is $3, 5, 7, 9$. For SESN model, we set the highest order of Hermite polynomial as 4 since it demonstrates the best performance. For the proposed models, we carefully set the lower and the upper bound of $\frac{l}{k}$ to set the size of filters (derived from equation (4.9)) of the first layer to be consistent with that of SESN, SDCF and DISCO. The range of each layer is shown in Figure 4.7a (black dashed lines). We set the highest order of the Gaussian derivative to be 1, since it demonstrates competitive performance yet uses a relatively fewer number of parameters. For all of the scale equivariant architectures, the number of filters for each scale group is one-quarter that of the standard UNet, except for the SEUNet , which is configured to match the number of parameters of the DISCO architecture. The colour normalisation method proposed in [102] is used to remove stain colour variation, before training. All models are trained on images at the original scale and then tested on multiple scales. Any scale augmentation such as random crop and then resize is not used in any of our experiments.

We implement the conventional UNet model and our proposed methods. The officially released source code of SESN, SDCF and DISCO layers is used in our experiments. All Models are implemented in PyTorch [86] and trained on one NVIDIA RTX 8000 GPU using the Adam optimiser [52] with weight decay of 10^{-6} to minimise the cross-entropy

loss. The training epoch is set as 50, and the initial learning rate for the Adam optimiser is set as 0.004 and then changed according to the 1cycle learning rate policy [95]. The batch size is 16 for training all models.

4.3.3 Results and Discussion

In this section, we report the overall and per-scale segmentation performance of the compared methods, followed by ablation studies to analyse the performance gain of our approach. The mean Intersection over Union (mIoU) is used to measure the segmentation performance of models. Examples of images, masks and segmentation maps generated by models can be seen in Appendix B.2.

Performance of each group of filters. Although we offer 3 strategies to generate the final segmentation maps from predictions, we first check the segmentation performance of each independent group of filters, as well as the proposed ensemble strategies. As shown in Figure 4.4, the best prediction shifts to the one with larger values, as the scale of images increases. Taking the CRAG dataset as an example, as the scaling factor increases, the highest mIoU scores for scales between [0.25-0.71], [0.84], [1-1.19], and [1.41-4.0] come from the branch of ^{1}, ^{2}, ^{3} and ^{4}, respectively (when ensemble strategies are not considered). This is consistent with our intuition that to capture the same information from enlarged images, filter sizes should also increase. Given a list of re-scaled images, we visualise segmentation maps that are generated by different

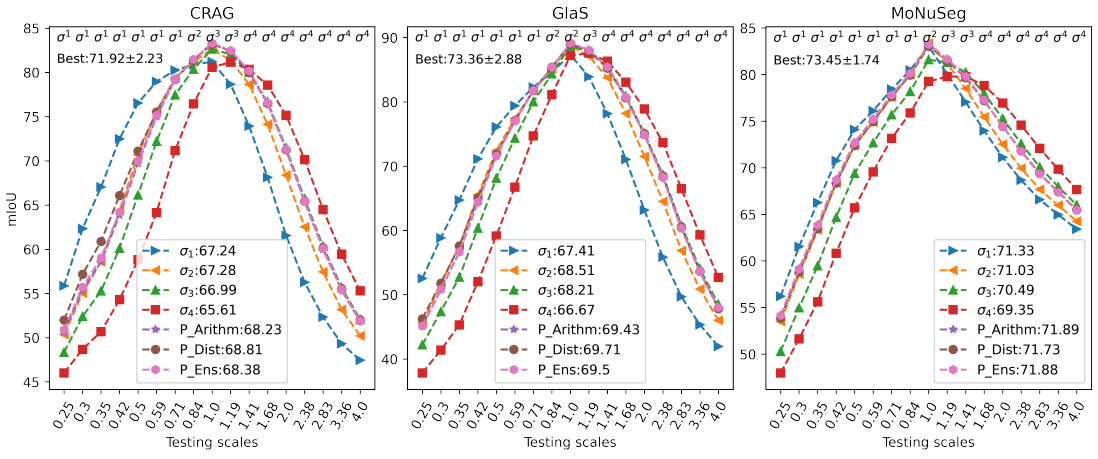


FIGURE 4.4: Per-scale mIoU score of each independent group of filters and their ensemble. The legend shows the mean of mIoU scores over all testing scales. Each dot denotes the mIoU score of 5-fold cross-validation. Here we use $_{k}$, k $1,2,3,4$, $_{1}$ $_{2}$ $_{3}$ $_{4}$ to denote different group of filters. The sequence of ^{i} shown on the top of each plot denotes the best-fit scale group that achieves the highest mIoU score for each specific testing scale. For each testing scale, we chose the highest mIoU from all four groups, calculating the mean std. (Standard Deviation) over all testing scales and then denoting it as 'Best' (shown in the top-left corner). i.e., the 'Best' represents the mean std. of mIoUs achieved by groups of filters shown on the top of the plot.

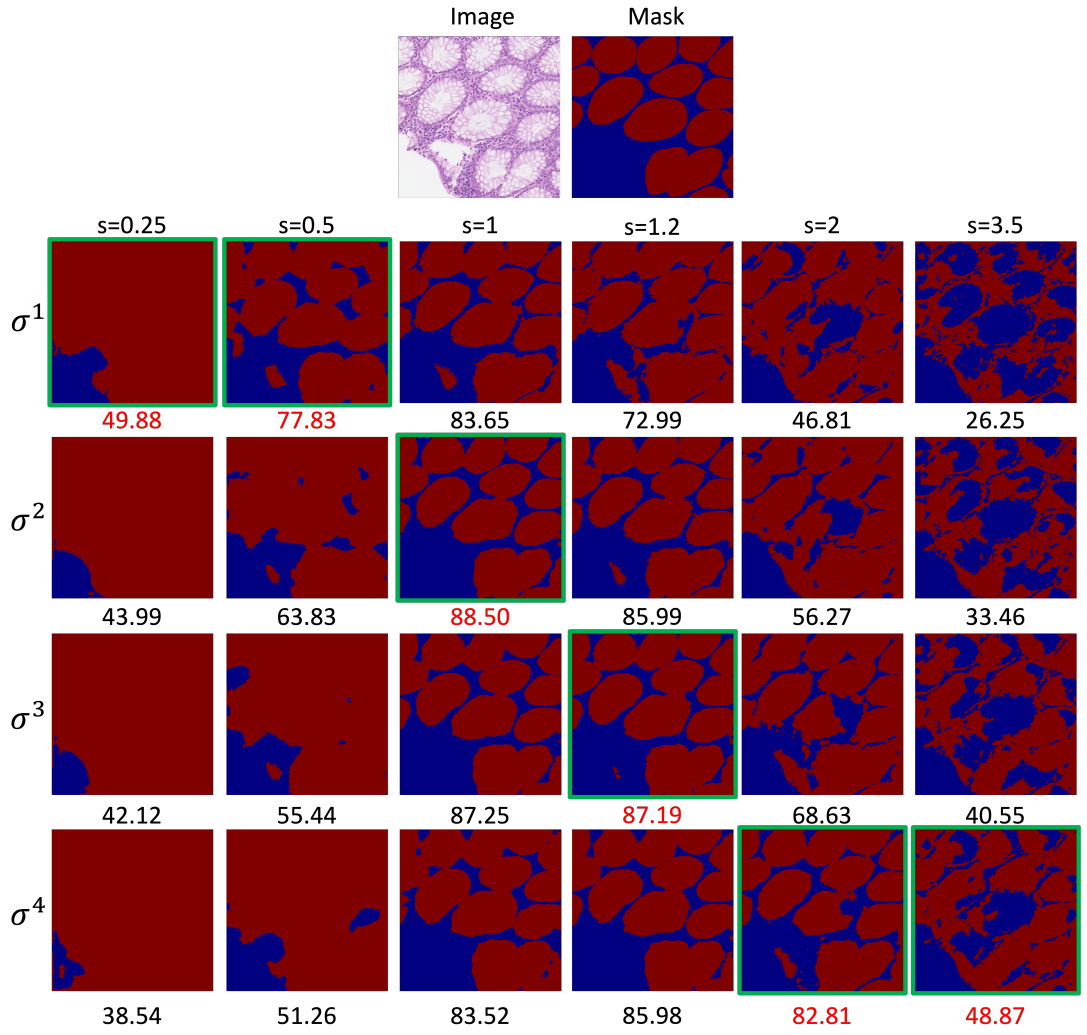


FIGURE 4.5: Per-scale prediction visualisation. The IoU score of each prediction is shown below the segmentation map. The ^{k} shown on the left indexes filters parameterised with different scale parameters ^{1} ^{2} ^{3} ^{4}. The re-scaling factors s are shown on the top of the 4 6 grid. The best match between images and filter scales are highlighted by a green box and red font (the highest mIoU score within each column). The image patch is randomly selected from the CRAG dataset.

scale groups to illustrate the shift of the match between filter scales and image scales. As shown in Figure 4.5, for consistent segmentation, up-scaled images correspond to larger filter scales and vice versa. The example shown in Figure 4.5 corresponds to curves shown in Figure 4.4, i.e., the same scale transform should be applied to filters when the image is zoomed in or zoomed out. In appendix B.1, we visualise the equivariance error of feature maps to demonstrate the effectiveness of lowering equivariance error by convolving images with multi-scale filters.

In terms of prediction aggregation strategies, all three methods outperform each single group of filters, from the perspective of all scales. When looking at a specific scale, however, all three strategies seem to fail to let the best branch contribute more but resulting in a performance lying between the best and the worst. Among these strategies, simply

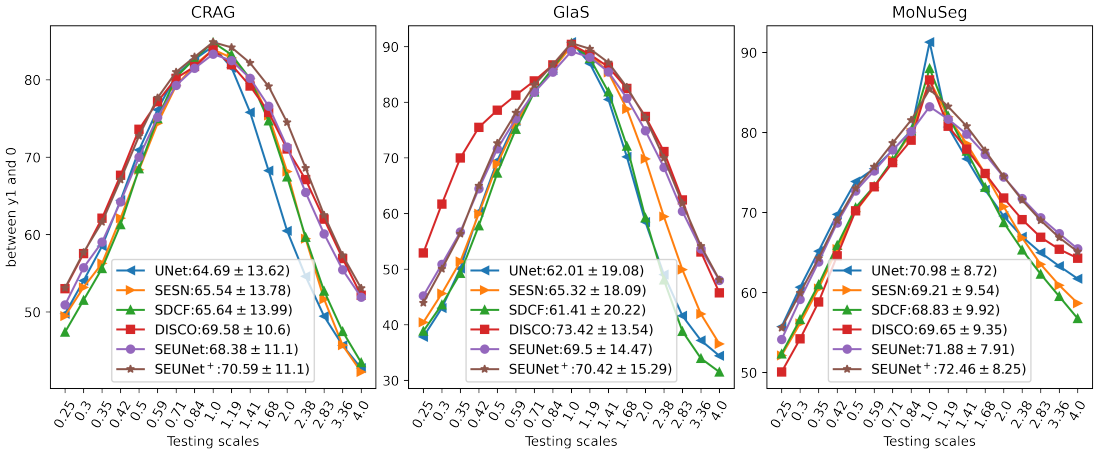


FIGURE 4.6: Comparison of the Per-scale mIoU score between models. The legend shows the mean S.E. of mIoU scores over all testing scales. Each dot denotes the mIoU score of 5-fold cross-validation.

averaging the prediction demonstrates the worst performance on the CRAG and the GlaS datasets. This suggests that mixing features of all scales equally without considering the possibility that scale-specific filters have different contributions to the prediction is not the optimal choice. This comparison validates the effectiveness of the P_Dist and P_Ens strategy. Although these strategies fail to pick up the most suitable scale for re-scaled inputs, Figure 4.4 indicates that the SEUNet has the potential of achieving a better scale generalisation performance if an appropriate prediction aggregation or prediction selection method is applied.

Comparison with CNN baseline and other equivariant methods. Figure 4.6 shows the per-scale test performance of different approaches on the CRAG, GlaS and MoNuSeg datasets. The performance of all models is similar when the test scale is close to the original training scale. However, as the test scale moves away from the training scale, the performance of the conventional CNN drops significantly, on the CRAG and GlaS datasets. Although the performance of SESN, SDCF, DISCO and the proposed method also drops as the test scale changes, these models look more robust than the CNN.

Table 4.1 compares models in terms of the averaged mIoU scores over different scales, the number of parameters, required GPU memory, and training speed. As observed from the table, the proposed SEUNet outperforms the conventional CNN (UNet) by 0.9, 3.69, and 7.49 points on MoNuSeg, CRAG and GlaS datasets respectively, while using just 2.2% of parameters of the UNet. This dramatic reduction in the number of parameters is owing to two reasons: 1) the weight sharing among scale groups, i.e., filters from different scale groups share the same combination coefficients . 2) learning combination coefficients of filter basis rather than the filter itself. As a result, this reduction shrinks the searching space of trainable parameters and may contribute to a lower risk of over-fitting. However, a model with fewer parameters may lead to a limited representation learning capability. As seen in the right plot of Figure 4.6, the

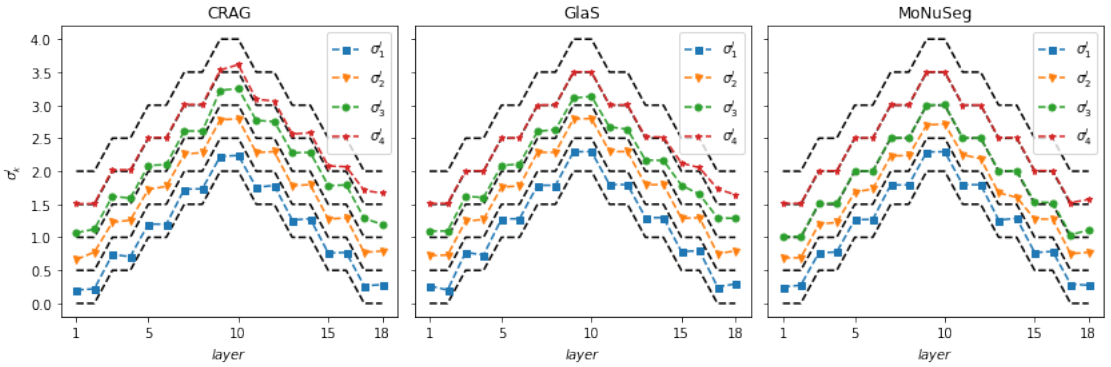
	UNet	SESN	SDCF	DISCO	SEUNet	SEUNet
Params (M)	28.95	9.66	9.66	1.81	**0.64**	1.81
GPU (GB)	4.69	5.09	5.09	**4.62**	5.06	8.52
Time (s)	**60**	296	293	207	374	675
MoNuSeg	70.98	69.21	68.83	69.65	71.88	**72.46**
	0.95	1.04	1.08	1.01	0.86	0.89
CRAG	64.69	65.54	65.64	69.58	68.38	**70.59**
	1.48	1.49	1.52	1.15	1.20	1.20
GlaS	62.01	65.32	61.41	**73.42**	69.50	70.42
	2.07	1.96	2.19	1.47	1.57	1.66

TABLE 4.1: The comparison of models in terms of the number of trainable parameters, required GPU memory, training time (seconds) per epoch, and segmentation performance (measured by mIoU). The GPU consumption and training time are measured with 512 512 inputs. The mIoU scores reported here are the mean S.E. (Standard Error) of 85 runs (5-fold cross-validation 17 testing scales). For SEUNet , the number of filters at each depth is 108, 216, 432, 864 and 1728. The fewest amount of parameters, training time and GPU memory of compared architectures, and the highest mIoU score achieved by models, are highlighted in bold.

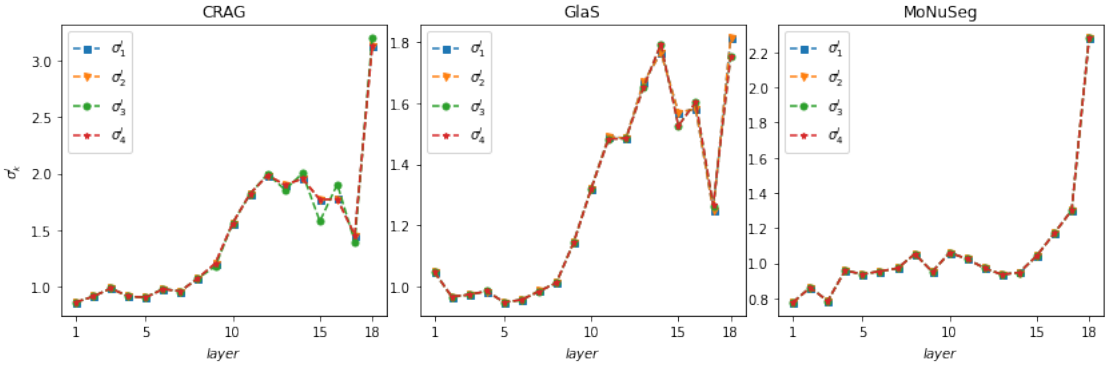
conventional UNet outperforms all the scale equivariant methods on the test set that is on the same scale as the training set. When compared with other scale equivariant counterparts, the SEUNet outperforms SESN and SDCF on all three datasets, while using just 6.63% of their required parameters and consuming almost the same amount of GPU memory. The SEUNet surpassed DISCO on the MoNuSeg dataset but shows inferior performance on the CRAG and GlaS datasets. After matching the number of parameters between SEUNet and DISCO by increasing the number of filters of the SEUNet, the result on the CRAG dataset turned over and SEUNet outperforms DISCO on the MoNuSeg dataset further. By checking the per-scale mIoU plot on the GlaS dataset that is shown in the middle of Figure 4.6, we see that the gap between SEUNet and DISCO mainly comes from the down-scaled test sets where DISCO yields higher mIoU scores. Looking back to Figure 4.4, nevertheless, the ^{1} branch of SEUNet seems to achieve similar performance as achieved by the DISCO on down-scaled versions of the dataset. Moreover, by picking up the best branch, the SEUNet is able to achieve a competitive performance (73.36 2.88 *v.s.* 73.42 1.47 achieved by DISCO, on the GlaS dataset). This again, highlights the necessity of proposing a highly effective aggregation method, collaborating with multi-scale convolutions.

4.4 Ablation Study

In section 4.2.3, we propose to constrain the value of $\frac{l}{k}$ in some non-overlapping ranges, to ensure that the constructed filters capture relevant patterns at different scales. Here, to verify the effectiveness of imposing range constraints and to trace the origin of



(A) σ_k^l is constrained under equation 4.8.



(B) σ_k^l is trained freely.

FIGURE 4.7: The σ_k^l values of filters in different layers. (A) The σ_k^l of models trained under range constraints. The black dashed lines show the lower and upper bounds a_k^l, b_k^l . (B) The σ_k^l of models trained without imposing range constraints.

	MoNuSeg		GlaS		CRAG	
Free	69.95	0.93	62.22	1.79	63.59	1.39
Fixed	72.03	0.86	69.23	1.70	68.18	1.32
Cons	**72.16**	**0.87**	**69.63**	**1.60**	**68.63**	**1.23**

TABLE 4.2: The mean S.E. of mIoU of SEUNets trained in different settings. "Fixed", "Free", and "Cons" denote that the scale parameters σ_k^ls are either fixed, trained without range constraint, or trained under the constraint of equation (4.8). The highest order N is set to 2 for all SEUNets.

the performance gain of the proposed method, we conduct the following two ablation experiments.

- **Fix σ_k^l values.** Instead of constraining σ_k^l to the range of a_k^l, b_k^l in equation (4.8), we fix the σ_k^l to be $\frac{a_k^l \ b_k^l}{2}$.
- **σ_k^l being trained freely. Where range constraints are removed.**

Table 4.2 summarises the performance achieved by models trained under different σ_k^l settings. The mIoU score reported in the table is the mean S.E. of the per-scale mIoU score obtained by P_Ens. As shown in the table, models trained with range constraint

slightly outperform models with $\frac{l}{k}$ being fixed and drastically surpass those trained completely freely, on all three datasets. Figure 4.7 shows the $\frac{l}{k}$ values of models trained under different settings. As can be seen in Figure 4.7b, allowing $\frac{l}{k}$ to be trained freely results in the case that multiple $\frac{l}{k}$s converge to the same value. This is detrimental to the feature representation of the model since the same $\frac{l}{k}$ means that the same scale of the generated filters, thus the resultant feature map is also the same (because the coefficient is shared between filters in different groups). Therefore, features are redundant and are not scale equivariant. This is the reason why the model trained without range constraint demonstrates the worst performance. The model trained with fixed $\frac{l}{k}$ values performs better than the freely trained one since $\frac{l}{k}$s are non-overlapping, thus multi-scale filters can be constructed to extract information from different scaled images. However, the manually selected $\frac{l}{k}$s may not be the optimal choice that fits the dataset best. Moreover, the optimal set of $\frac{l}{k}$s may vary from dataset to dataset. This motivated our choice in equation (4.8) to train $\frac{l}{k}$s to remain in disjoint intervals. As observed from Figure 4.7a, $\frac{l}{k}$s trained under constraint deviate from fixed values. And also, the learned $\frac{l}{k}$s vary between datasets. That indicates different datasets may favour different scale parameter settings, allowing scale parameters to be trainable will thus release the capability of the model to search for the optimal one.

Another benefit of imposing range constraints on $\frac{l}{k}$ is to reduce the computational complexity of the model. In our experiments, we observe that the model trained with range constraints required only $\frac{1}{3}$ the training time of the freely trained one. Because the filter of smaller size is less computationally intensive when performing convolution operations. For example, we observe that the maximum $\frac{l}{k}$ values can reach up to 9.99 during training, resulting in the corresponding filter sizes of 41 41 (calculated from equation (4.9)). Therefore, the amount of computation required by a 41 41 filter is 5.82 times that of a 17 17 filter, when convolving with an image.

4.5 Summary

In this chapter, we propose a Scale-Equivariant UNet (SEUNet) to address the challenge of generalising neural network segmentation on histopathology images to unseen scales. The motivation behind this is to achieve robust segmentation for histopathology images of unknown magnification level. To fulfil this objective, we parameterise multi-scale filters by linearly combining groups of Gaussian derivative filters. The constructed filters are then used to learn scale-space representations that have a built-in scale-equivariant property. We constrain filter scales to be both trainable yet cover disjoint ranges. This is useful for finding dataset-adapted scale parameters. The extensive experimental results on two public datasets demonstrate that the proposed SEUNet achieves state-of-the-art performance.

However, the proposed framework has the following shortcomings that we seek to overcome in the future:

- Since we learn the Gaussian derivatives during training, these derivatives should be updated after each round of $\frac{l}{k}$ updating, which is more computationally expensive than using a pre-calculated filter basis. In the future, some effort will be put to optimise this step to prevent the long processing time of training.
- We currently use a unified predefined range to restrict the values of $\frac{l}{k}$, while ignoring the fact that different datasets may favour different range settings. Therefore, more range constraints will be explored to enable improved performance, or furthermore, make range constraints also trainable.
- Although the proposed ensemble strategies P_Arithm, P_Dist and P_Ens are effective for aggregating probabilities to generate a final prediction, they fail to pick up a group of filters that fits a specific scale best. Therefore, it would be interesting to explore other strategies that are able to pull out the most accurate prediction from multiple parallel branches.

Note that variations in images go beyond scale transformation, and also include rotation transformation. In the next chapter, we integrate rotation equivariance into CNNs to release neural networks' representation capacity, thus further enhancing generalisation performance.

Chapter 5

Joint Rotation-Scale Equivariance for Stabilised Segmentation

5.1 Motivation and Contribution

CNNs utilise a weight-sharing strategy to guarantee that a translation of the input will result in a corresponding translation of the features. This property, known as translation equivariance, is an inherent property of the CNN and removes the need to learn features at all spatial locations, resulting in a significant reduction in the number of learnable parameters. In certain image analysis applications where orientation does not affect the semantic meaning of images, it is beneficial to extend this property of equivariance beyond translation to rotation. By doing so, CNNs are empowered to pick up features regardless of the angle shift of images. Another desirable property of CNNs is scale equivariance, which can be beneficial for CNNs to recognise features regardless of the scale variation.

Recent work has shown that exploiting rotation and scale equivariance in convolutional neural networks can improve models' generalisation performance [20] and sample efficiency [71, 121]. Domains that benefit most from exploiting both equivariance are those where the image itself lacks canonical orientation and scale, and where the number of samples is limited. These are features of digital histopathology images, where localised patterns can appear in any orientation (see Figure 5.1 for example), and the size of cells and tissues covary with different choices of objective magnifications used to digitise specimen slides. In medical image-related tasks (which generally have very limited samples), one can use offline data augmentation for better generalisation, but this is at the cost of increasing the number of training samples. Previous work [9, 39, 40, 71, 103] shows that incorporating rotation equivariance into CNNs leads to better performance on the classification, detection, and segmentation of histopathology images. In chapter 4, we demonstrate that introducing scale equivariance into CNNs improves models'

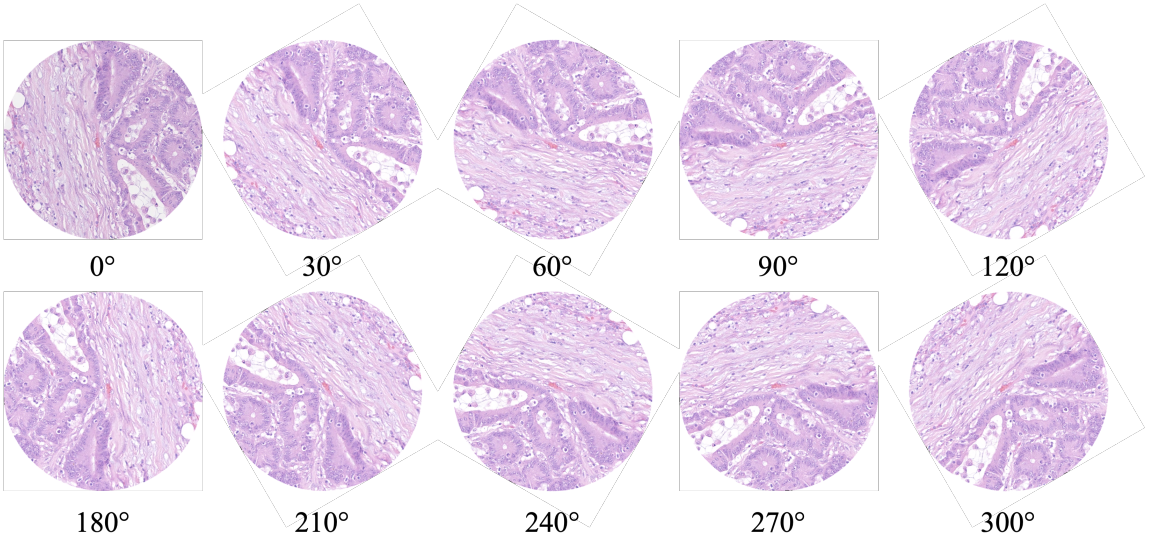


FIGURE 5.1: Cropped central regions from a H&E stained image. Each orientation is equally as likely to appear.

segmentation performance when tested on histopathology images that are presented in unseen scales. However, all of these works either consider rotation symmetry or scale symmetry, which did not fully extract CNNs' potential for joint rotation-scale equivariance.

In this chapter, we first identify the two reasons why the generalisation performance of conventional CNNs is limited, on histopathology image analysis: 1) a part of the parameters of filters are trained to fit rotation transformation, thus decreasing the capability of learning other discriminative features; 2) fixed-size filters trained on images at a given scale fail to generalise to those at different scales. To deal with these issues, we extend the SEUNet (proposed in chapter 4) and then propose the Rotation-Scale Equivariant Steerable Filter (RSESF), which utilises filter steerability and Gaussian scale-space theory to parameterise convolutional filters, resulting in an equivariant layer that is stable to rotation and scale variations. The RSESF contains copies of filters that are linear combinations of Gaussian derivative filters, whose direction is controlled by directional derivatives and whose scale parameters are trainable but constrained to span disjoint scales in successive layers of the network. The main contributions of this chapter are summarised in Figure 5.2 and listed as follows:

- We extend the scale equivariant only SEUNet to a joint rotation and scale equivariant framework, by introducing the directional derivatives.
- The constructed framework achieves a higher degree of weight sharing, greatly beyond CNNs which only have translation equivariance.

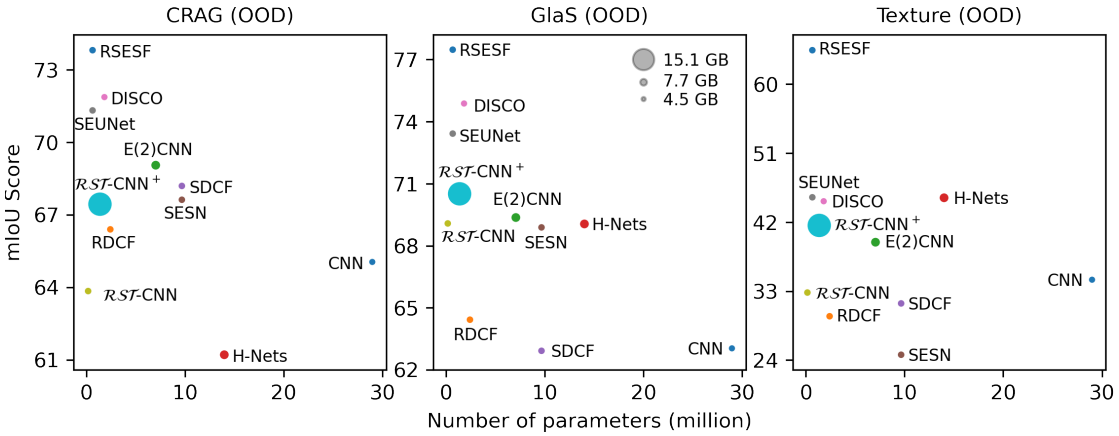


FIGURE 5.2: **Model Size vs. Segmentation Performance vs. GPU Requirement.** Models evaluated on out-of-distribution setting. Our RSESF outperforms others on all of three datasets. Notably, the RSESF achieves state-of-the-art mIoU but is much smaller and more GPU efficient. The larger the diameter of the marker, the larger the GPU memory required for model training.

- By decoupling convolutions between rotation channels, RSESF possesses the flexibility of being trained with only one rotation channel but introducing other rotation channels during inference as needed. This not only guarantees generalisation performance but also enables memory-efficient training.
- We demonstrate state-of-the-art generalisation performance across multiple datasets on image segmentation tasks.

5.2 Methodology

Rotation-Equivariance. For a rotation operator T defined on a function (image) represents rotating an image by and for a family of feature mapping operators (typically a convolutional layer), rotation equivariance means that the rotation transformation should commute with the feature mapping operation according to

$$T f \quad T \quad f \ , \quad 0,2 \ , \tag{5.1}$$

where denotes some feature map operators within the same family that operates on the image rotated by a angle of .

Joint Rotation-Scale Equivariance. An operator $T_{,s}$ defined on a function (image) represents rotating and re-scaling an image by an angle of and by a factor of s. For a family of feature mapping operators, joint rotation-scale equivariance means that the transformation should commute with the feature mapping operation according to

$$T_{,s} f \quad T_{,s} \quad f \ , \quad 0,2 \ , s \quad 0. \tag{5.2}$$

When $\theta = 0$, the transformation $T_{\theta,s}$ degenerates to the scaling operator, and equation (5.2) denotes scale equivariance. When $s = 1$, the transformation $T_{\theta,s}$ degenerates to the rotation operator, and equation (5.2) denotes rotation equivariance.

5.2.1 Rotation-Scale Steerable Gaussian Derivative Filter Basis

In chapter 4, we linearly combine the 2D Gaussian derivatives,

$$G^{i,j}(x,y;\sigma) = \frac{\partial^{i+j}G(x,y;\sigma)}{\partial x^i \partial y^j} = \frac{\partial^i G(x;\sigma)}{\partial x^i}\frac{\partial^j G(y;\sigma)}{\partial y^j},\ i,j \geq 0,\ i+j \leq N \tag{5.3}$$

to construct filters. Order N refer to the highest order of derivative used in equation (5.3). These basis elements capture variations along the x or y directions. To make filter basis able to capture features presented in arbitrary orientation, we define filter basis elements $G^1_\theta(x,y;\sigma)$ rotated by angle θ:

$$\begin{aligned} G^1_\theta(x,y;\sigma) &= \cos\theta\frac{\partial G(x,y;\sigma)}{\partial x} + \sin\theta\frac{\partial G(x,y;\sigma)}{\partial y} := \left(\cos\theta\partial_x + \sin\theta\partial_y\right) G(x,y;\sigma) \\ &:= \cos\theta G^1_0(x,y;\sigma) + \sin\theta G^1_{\frac{\pi}{2}}(x,y;\sigma) \end{aligned} \tag{5.4}$$

with $\cos\theta, \sin\theta$ as interpolation functions used in steering theorems [30]. Using equation (5.4), we can simultaneously control the size and orientation of convolutional filters by manipulating σ and θ parameters of their filter basis. The benefit of using a steerable filter is that it avoids the interpolation artefacts produced by directly rotating the filter. The $N = 2$ filters are $(\cos\theta\partial_x + \sin\theta\partial_y)^2 G(x,y;\sigma)$ as shown in Appendix C.1.

5.2.2 Filter Construction.

We denote the filter basis rotated by θ as $G^{i,j}_\theta(x,y;\sigma)$. Then we parameterise the proposed Rotation-Scale Equivariant Steerable Filter as a linear combination of directional Gaussian derivative filters:

$$F^l_k(c^l,x,y,\sigma^l_k,\theta_r;c^{l-1}) = \sum_{i,j\geq 0}^{i+j\leq N} \alpha^l_{i,j,c_l,c^{l-1}} G^{i,j}_{\theta_r}(x,y;\sigma^l_k),\ 1 \leq k \leq \gamma,\ \theta_r = \frac{2\pi r}{R}, 1 \leq r \leq R \tag{5.5}$$

where $l \geq 1$ is the layer index, c^{l-1} and c^l are the channel indices of the input and output of the l^{th} layer, k is the index for the γ scales and r is an index representing an orientation out of R orientations, in a layer. The expansion coefficients $\alpha^l_{i,j,c^l,c^{l-1}} \in \mathbb{R}$ and scale parameter σ^l_k are learnable. The filter F^l_k constructed from equation (5.5) is of the dimension $[C^l, C^{l-1}, R, h^l_k, w^l_k]$, which has R rotation channels. (h^l_k, w^l_k) denotes the spatial size of the filter, which is controlled by the scale parameter σ^l_k, i.e., $h^l_k = w^l_k = 2\lceil 2\sigma^l_k \rceil + 1$. We create γ groups of filters $[F^l_1, \cdots, F^l_\gamma]$, with each group sharing

the same expansion coefficients $^{l}_{i,j,c^l,c^{l\ 1}}$, but with different scale factors $^{l}_{k}$s to ensure scale equivariance. Within each scale, R rotation channels at angular resolution $\frac{2}{R}$ implement discrete rotation equivariant filters. Figure 5.3 illustrates the construction of the filter. We demonstrate that Gaussian derivative filters are equivariant to scale transformation of images, in section 4.2.1.1, and provide evidence in Appendix B.1 to support our statement. For joint rotation-scale equivariance of directional Gaussian derivative filters regarding the transformation of images, the equivariance error plots are shown in Appendix C.3 as evidence. We then describe feature extraction across scales in parallel.

5.2.3 Equivariant Convolution

For the first layer, we convolve the image with each group of filters in parallel. For channel c^1, 1 c^1 C^1 at scale k and rotation r, the convolution f^1_k F^1_k f^0

$$
\begin{aligned}
f^1_{k,c^1,r}\ x,y \quad & F^1_k\ c^1,x \quad x_0,y \quad y_0,\ ^1_k,\ _r;c^0\ f^0_{c^0}\ x_0,y_0 \\
& {\scriptstyle c^0\ ,\ ,\ x_0,y_0} \\
& {\scriptstyle i\ \ j\ \ N} \\
& \quad {\scriptstyle i,j\ \ 0\ \ c^0\ \ ,\ ,\ x_0,y_0}\ ^1_{i,j,c^1,c^0}G^i_{\ r}\ x;x_0,\ ^1_k\ G^j_{\ r}\ y;y_0,\ ^1_k\ f^0_{c^0}\ x_0,y_0
\end{aligned}
\tag{5.6}
$$

gives the output f^1_k $^{C^1\ \ R\ \ H\ \ W}$ for each of the scales.

For subsequent layers l 2 , the feature map from each scale group is only passed to the corresponding scale also indexed by k, i.e., f^l_k F^l_k $f^{l\ 1}_k$. Inside each scale group, for each output channel, feature maps of multiple orientations from the previous layer are independently convolved with multi-orientated filters and then summed over orientation channels:

$$
f^l_{k,c,r} \quad {\scriptstyle d\ \ 1\,r\ \ 1}^{\ C^{l\ 1}\ \ R}\ F^l_{k,c,d,r} \quad f^{l\ 1}_{k,d,r}\ ,\ \ c \quad 1,\ \ \ ,C^l\ ,k \quad 1,\ \ \ ,\ \ ,\ \ \tag{5.7}
$$

where r and r are the indices of the rotation channel for the filter and feature maps, respectively. The sum over orientation channels renders transformations between hidden layers invariant to rotations. Note, features from other orientation channels r 1, , R , r r of the input $f^{l\ 1}_{k,c}$ are not involved in the calculation of $f^l_{k,c,r}$, which means no orientation information is mixed through convolution. This gives the constructed network flexibility to adopt different numbers of rotation channels between the training and testing phases. This would thus enable the network to reduce the demand on GPU memory required for training while maintaining performance during inference. We will describe this benefit later.

5.2.4 Model Training

Decoupled convolution between rotation channels enables memory efficient training. Within each scale group, equation (5.6) indicates that the input image is convolved with R copies of rotated filters, separately. Equation (5.7) shows that the input of each orientation channel of hidden layers is also individually convolved with rotated filters to generate feature maps. Since there is no inter-rotation interaction between rotation channels, the information flow is independent across rotation channels. Therefore we

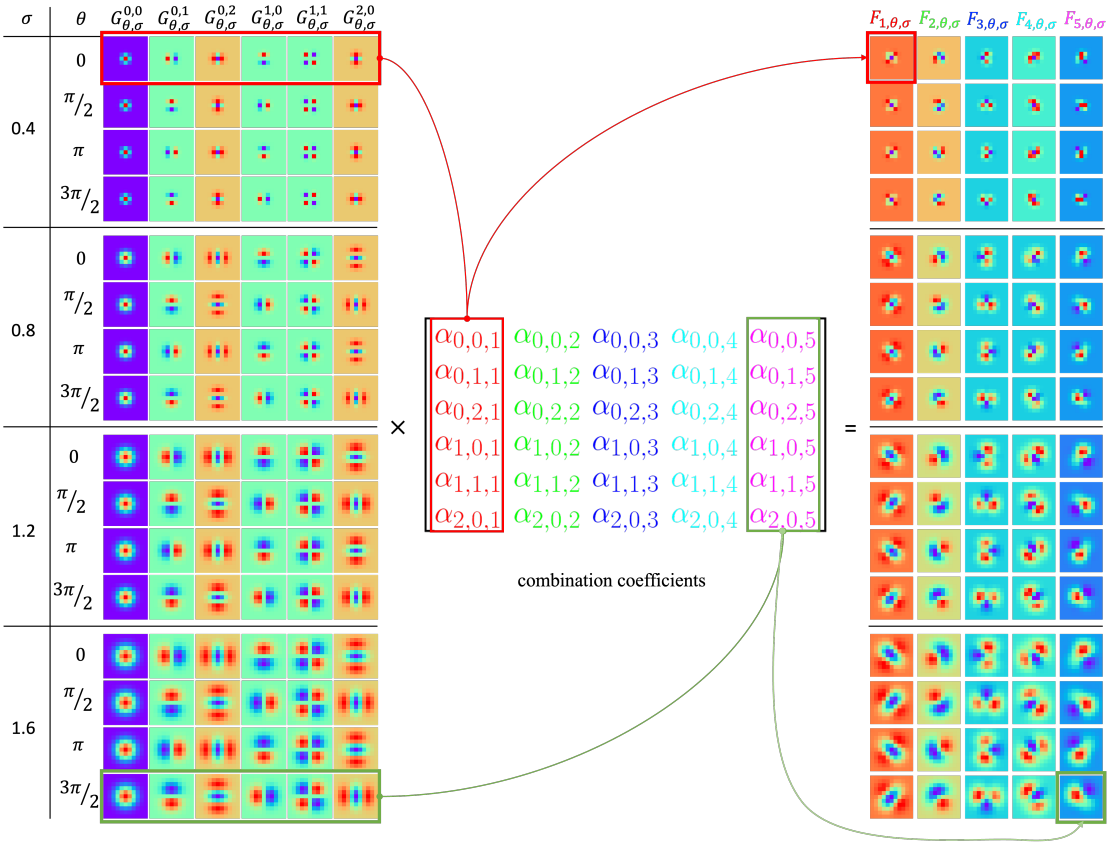


FIGURE 5.3: Here we depict the construction process of an RSESF filter (with 4 scale channels, 4 rotation channels, 1 input channel and 5 output channels) as matrix multiplication. The highest order of Gaussian derivative used here is 2. **Left.** Gaussian derivative filter basis $G^{i,j}$, parameterised by different and values from top to bottom, and different orders i, j (with respect to and $\overline{2}$ respectively) from left to right; **Middle.** Learnable linear combination coefficients $_{i,j,c}$, where c is the index of output channel. Each column vector is shared between scales, and it determines the shape of the final constructed filter. **Right.** The constructed multi-scale, multi-orientation filters (filters vary in shape from left to right, as learnable coefficients (column vectors) involved in the calculation are different). The red/green lines that connect the Gaussian derivative bases (left), combination coefficients (middle), and constructed filter (right) show the process of equation (5.5), i.e., the dot product between the Gaussian derivative bases and combination coefficients produces the convolutional filter. Note, for visualisation purposes, we set values to $0.4, 0.8, 1.2, 1.6$. However, they are allowed to be tuned in disjoint ranges as described in equation (5.8).

are allowed to train the network within only one rotation channel, but then other R-1 rotation channels after training are created to reduce the model's orientation sensitivity as needed. The filters of newly created rotation channels are guaranteed to be in the same shape and scale as the trained one, as the expansion coefficients are shared in rotation and scale dimensions. The reduction of the number of rotation channels reduces GPU memory consumption by a factor of R during the training, as other R-1 feature maps do not need to be stored in the memory in the back-propagation calculations. This translates into a two-fold advantage: firstly, a larger number of filters can be used, thus increasing the feature representation capability of the network; secondly, RSESF can be trained in GPU resource-limited settings, thus greatly increasing its applicability. Moreover, the GPU memory released from the rotation channel can compensate for the memory needed for using a larger batch-size, to achieve more stable training. Setting different number of rotation channels R during inference is discussed in section 5.4.2.

Parallel Training between Scales. We adopt the strategy proposed in section 4.2.3 for simultaneously training filters that are at different scales. The $^{l}_{k}$s remain in disjoint intervals, guaranteeing scale equivariance,

$$^{l}_{k}\ x \qquad \frac{a^l_k \quad b^l_k}{2}\tanh x \qquad \frac{a^l_k \quad b^l_k}{2}, \ a^l_k \quad b^l_k \quad 0, \ a^l_{k\ 1} \quad a^l_k, \ b^l_{k\ 1} \quad a^l_k, \tag{5.8}$$

where a^l_k and b^l_k are upper and lower bounds for the scale parameters of the filters at the l^{th} layer and the k^{th} group. x is a trainable real variable. At the last layer (the L^{th} layer)

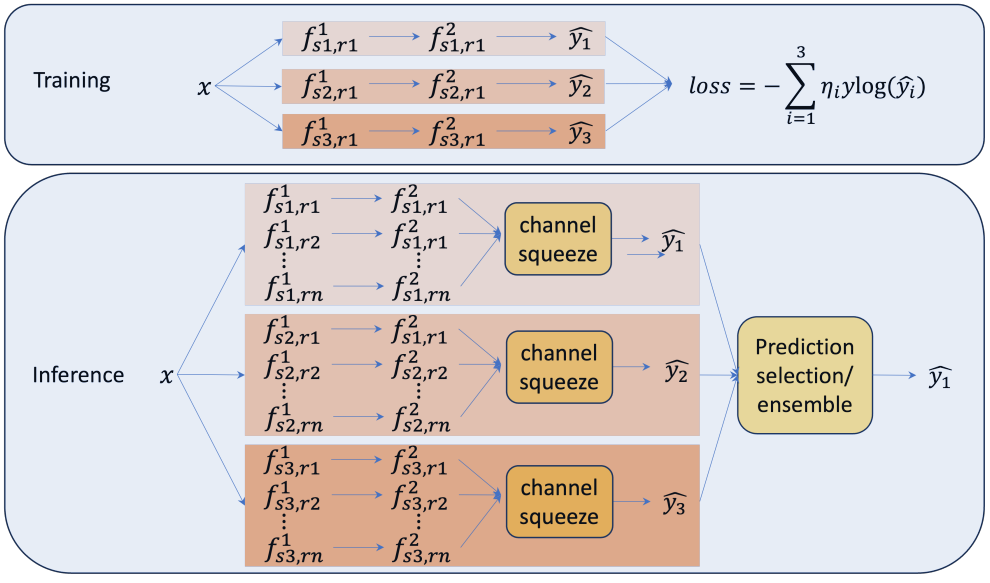


FIGURE 5.4: A two-layer example of training *v.s.* inference of the RSESF. The subscripts si and rj denote the index of scale channels and rotation channels, respectively. 'channel squeeze' reduces the dimension of feature maps by squeezing the rotation channel. Different strategies of 'channel squeeze' are discussed in section 5.4.1.

of the neural network, for a K-class segmentation task, a $K \times C^L \times 1 \times 1$ convolutional filter is used to squeeze the feature map $f_k^L \in \mathbb{R}^{C^L \times 1 \times H \times W}$ (the 1 in the second dimension means we only use 1 rotation channel for training) into $f_k \in \mathbb{R}^{K \times H \times W}$, followed by a softmax function that maps the f_k to K probability maps $\hat{y}_{k,c} \in \mathbb{R}^{H \times W}$, $c \in \{1, \cdots, K\}$, which is then used to calculate the combined cross-entropy loss per-pixel

$$\mathcal{L} = -\sum_{c=1}^{K}\sum_{k=1}^{\gamma} \widetilde{\eta}_k y \log(\hat{y}_{k,c}),\ \widetilde{\eta}_k = \frac{\eta_k}{2} + \frac{1}{2\gamma},\ 0 \leq \eta_k \leq 1,\ \sum_{k=1}^{\gamma} \eta_k = 1,\ \sum_{k=1}^{\gamma} \widetilde{\eta}_k = 1. \quad (5.9)$$

where $\widetilde{\eta}_k$ is a trainable rectified weighting factor to characterize the importance of the k^{th} scale. $\widetilde{\eta}_k$ is bounded in $[\frac{1}{2\gamma}, \frac{\gamma+1}{2\gamma}]$, which ensures that each scale contributes to the training. Note that for the same network, the feature map f_k^L is of the dimension $[C^L \times 1 \times H \times W]$ for training, but is of the dimension $[C^L \times R \times H \times W]$ in the inference phase. Therefore, in the inference phase, we max-pool the f_k^L over the rotation channel to squeeze it into $(f_k^L)^{\max} = \max_{1 \leq r \leq R} f_{k,r}^L$, $(f_k^L)^{\max} \in \mathbb{R}^{C^L \times 1 \times H \times W}$, so $(f_k^L)^{\max}$ can be further squeezed by the following 1×1 convolutional layer. Other ways of dimension reduction are explored and compared in section 5.4.1. Figure 5.4 shows the difference between the training and the inference phase of the RSESF.

5.3 Experiments and Results

5.3.1 Datasets

Histopathology datasets. We use CRAG [3] and GlaS [94] datasets, which are introduced in section 3.3.1, for evaluating our method.

Texture dataset. Alongside evaluating models on histopathology datasets that have built-in rotation symmetry properties, we create a synthetic texture segmentation dataset made from directional texture images to analyze the effectiveness of the proposed method on out-of-distribution testing. In detail, we customize masks to each contain five regions using the online Prague Texture Segmentation Data Generator[1]. Then five types of pure texture images (160 unique texture patches per class) from the Kylberg Texture Dataset (un-rotated textures) [59] are selected to create mixed mosaic images with boundaries aligned with mask shapes. Note that for each type of texture, all of the 160 pure texture patches are presented in only one orientation. We finally generate 720 mosaic-mask pairs, with each mosaic image containing 5 classes of textures. Figure 5.5 shows examples of pure texture images and a synthetic texture mosaic with the mask.

Rotation and scale augmented test set. For all datasets, to compare the generalisation capability (the ability to segment images that are presented in unseen scales and orientations) of models, we randomly rotate and re-scale the entire image of the initial test

[1] https://mosaic.utia.cas.cz/

set four times to create a new test set that has more orientation and scale variations for out-of-distribution testing.

Evaluation regime. Two criteria are used to evaluate the models' performance.

- 1) In-Distribution (ID) test, i.e., the training images are randomly rotated (by an angle uniformly distributed on $0, 2\ $) and re-scaled (by a factor uniformly distributed on [0.5, 4]). Trained models are then evaluated on an augmented test set.
- 2) Out-Of-Distribution (OOD) test, the training images are fed into models *without* any rotation and scale augmentation but are randomly horizontally and vertically flipped. Trained models are then evaluated on an augmented test set.

For each dataset, 5-fold cross-validation is used to measure the performance. Figure 5.6 shows the procedure of ID and OOD evaluation regimes. Note that the configuration of models is the same for ID and OOD evaluation.

5.3.2 Model Settings and Implementation Details

We use the UNet architecture as a backbone and adopt the conventional convolutional layer as well as other types of equivariant convolution layers (including RDCF [16], E(2)CNN [111], H-Nets [115], SDCF [130], SESN [96], DISCO [98], SEUNet [121], $\mathcal{RST}$-CNN [34], and the proposed RSESF) to investigate 11 models. For the UNet with norm CNN layers, the number of convolutional channels at each depth is 64, 128, 256, 512, and 1024. For a fair comparison, we keep the total number of channels of equivariant UNet variants in line with the conventional UNet. In detail, we split convolutional channels into N_f filter channels, scale channels, R rotation channels, and guarantee that N_f R is the same for all models. The values of N_f, and R of each model are summarised in Table 5.1. For SEUNet and RSESF, we set the highest order of the Gaussian derivative to be 1. We carefully set the scale factors for SESN and SDCF, so that their receptive field sizes are consistent with SEUNet and RSESF. For GlaS and CRAG datasets, the colour normalisation method proposed in [102] is used to remove stain colour variation, before training. All models are 1) trained with random rotation and scale augmentation for the In-Distribution test, and 2) trained on images at the

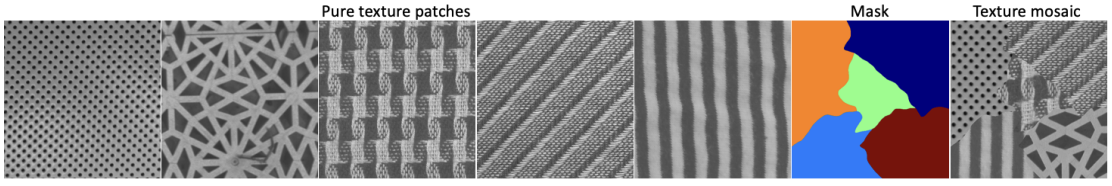


FIGURE 5.5: Pure texture images and generated texture mosaic with the corresponding mask.

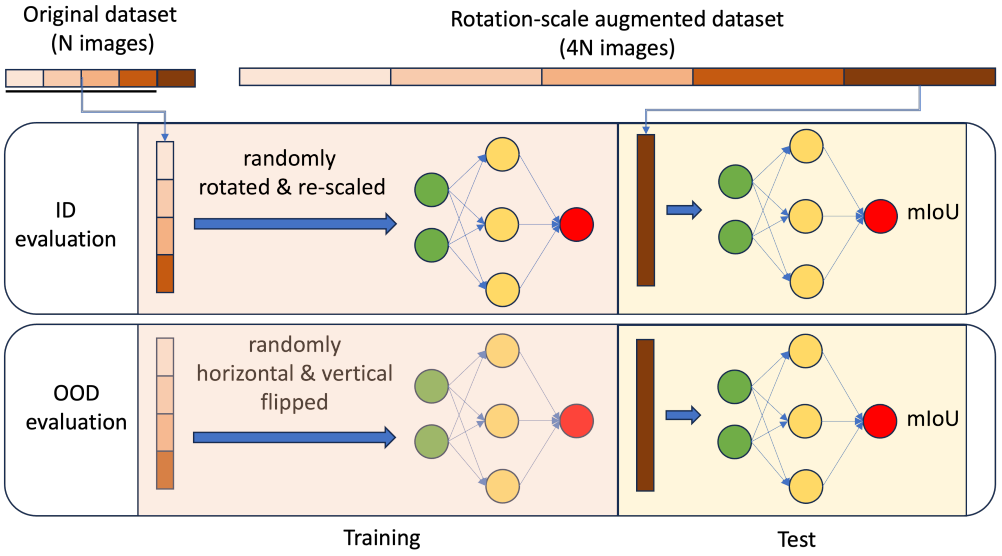


FIGURE 5.6: ID *v.s.* OOD Evaluation regimes.

original orientation and scale, without rotation and scale augmentation (only randomly horizontal and vertical flip augmentation is used).

We implement the conventional UNet model, SEUNet, and our RSESF. We use the e2cnn library[2] to build E(2)CNN and H-Nets. For RDCF, SDCF, SESN and $\mathcal{RST}$-CNN, the code associated with Gao et al. [34] is used to build equivariant layers. All Models are implemented in Pytorch [86] and trained on one NVIDIA RTX 8000 GPU (45GB memory) using the Adam optimizer [52] to minimize the cross-entropy loss. To fully utilize the GPU memory for efficient training, we set the batch size to 6 for $\mathcal{RST}$-CNN and 16 for all the other models. The learning rate, training epochs, and weight decay coefficient are optimized by Random Search [13].

5.3.3 Results

We organise the experimental results by evaluation criteria.

ID testing. Table 5.1 summarises the comparison between models in terms of segmentation performance under In-Distribution evaluation, the number of trainable parameters, and the amount of GPU memory required for training. As shown in Table 5.1, all of the equivariant models outperform the conventional CNN, except the H-Nets, on the GlaS and CRAG datasets. This indicates that they are able to extract useful representations for segmentation, even with much fewer parameters. In detail, RSESF outperforms CNN, but its number of parameters is just 2.21% of CNN. The E(2)CNN ranked at the top among other methods, but the number of its parameters is 10.98 times that

[2]https://github.com/QUVA-Lab/e2cnn

of RSESF. Other scale and rotation equivariant models also achieve competitive performance but with more parameters, when compared with RSESF. The poor performance of the $\mathcal{RST}$-CNN model, a UNet with jointly rotation-scale equivariant convolutional layers, suggests that it is characterised with too few filters to learn adequate features. By tripling the number of filters we create $\mathcal{RST}$-CNN with improved performance, but at the cost of tripling the amount of GPU required for training. This memory overhead is expensive; thus a trade-off between performance and computation resources needs to be considered. The proposed RSESF, however, is not constrained by the GPU memory while training. This memory efficiency during training in our RSESF stems from parallel training and decoupled convolution between rotation channels. The performance gain from SEUNet to RSESF, i.e., from one rotation channel to 8 rotation channels, is marginal. We attribute this to the presence of rotation augmentation during training, which makes SEUNet able to memorise features shown in different orientations and thus achieve a close mIoU score to RSESF. Similarly, E(2)CNN achieves the best performance. This is due to the fact that on one hand, E(2)CNN is rotation equivariant and on the other hand, it memories features in various scales because of scaling augmentation.

OOD testing. In terms of OOD testing, which is designed to evaluate models' generalisation capacity to rotation and scale variations, the proposed RSESF demonstrates the best performance on all three datasets, as shown in Figure 5.7, although it has the second least number of parameters (reported in Table 5.1). The results of OOD and ID evaluations differ significantly due to the presence or absence of rotation and scaling augmentation. In the absence of any augmentation, models enjoying equivariance

TABLE 5.1: Model comparison for histopathology datasets in terms of the number of parameters, GPU memory required for training, and mean S.E. of mIoU for In-Distribution evaluation. Columns N_f, , R denote the number of filters, scale channels, and rotation channels in the first layer of the UNet, respectively. Note that for RSESF, "R 1,8" denotes that only 1 rotation channel is used during training but 8 rotation channels are used during testing. The total number of channels is matched between the $\mathcal{RST}$-CNN and RSESF. The top-3 performances on each dataset are highlighted in red, green, and blue, respectively.

Filter Type	Param	GPU	N_f		R	GlaS		CRAG		Texture	
CNN	28.95	4.48	64	1	1	81.02	0.18	73.94	1.05	99.12	0.01
RDCF	2.42	5.07	8	1	8	85.54	0.24	81.65	0.59	99.09	0.05
E(2)CNN	7.03	7.65	8	1	8	88.54	0.18	84.49	0.44	99.57	0.01
H-Nets	13.98	7.65	64	1	1	77.49	0.79	69.17	1.02	98.35	0.07
SDCF	9.66	5.09	16	4	1	84.52	0.38	80.95	0.62	99.10	0.03
SESN	9.66	5.06	16	4	1	83.88	0.38	79.44	0.58	98.61	0.02
DISCO	1.81	4.62	16	4	1	86.44	0.25	83.4	0.66	99.05	0.02
SEUNet	0.64	5.05	16	4	1	85.45	0.32	82.64	0.59	98.47	0.03
$\mathcal{RST}$-CNN	0.15	5.06	2	4	8	81.18	0.54	74.21	0.72	94.97	0.23
$\mathcal{RST}$-CNN	1.36	15.12	6	4	8	85.38	0.39	81.63	0.31	98.83	0.04
RSESF	0.64	5.06	16	4	1,8	85.85	0.37	83.11	0.60	98.48	0.03

properties demonstrate higher performance. Models without rotation/scale equivariance properties are not able to learn patterns presented in arbitrary orientations and scales, and thus can not recognise them after being trained without data augmentations. In contrast, equivariant models are able to recognise patterns without orientation and scaling augmentations. In detail, 8, 9 and 6 out of 10 equivariant models outperform the conventional CNN on the CRAG, GlaS, and texture datasets, respectively. On histopathology datasets, the top 3 models are the rotation-scale equivariant model (RSESF 1^{st}), and the scale equivariant models (DISCO 2^{nd}, SEUNet 3^{rd}). It is noted that the 3^{rd} ranked model on the texture dataset is a continuous rotation equivariant model (H-Nets). Such differences in results among datasets are due to the characteristics of datasets. For example, glands are equally as likely to appear in every orientation, on the CRAG and GlaS datasets. However, the texture of the synthetic mosaic is usually associated with a specific directionality. Therefore, it is reasonable that an H-Nets achieves competitive performance on the augmented texture test set as its inferior prediction on re-scaled images is compensated by its higher rotation equivariance. Notably, by absorbing rotation equivariance into SEUNet, the mIoU of RSESF on the CRAG, GlaS, and texture datasets are boosted by 2.50, 4.04, and 19.22 points, respectively. That highlights the superiority of joint rotation and scale equivariant models.

The $\mathcal{RST}$-CNN , a counterpart of the joint rotation and scale equivariant method, ranked 4^{th} and 5^{th} on the GlaS and texture datasets. We attribute this limited segmentation performance, as in the ID case, to the inadequate number of filters (N_f 6), thus leading to inferior learning capacity. However, we are not able to train a $\mathcal{RST}$-CNN which has more filters, due to the GPU memory constraint.

Orientation-Scaling sensitivity visualisation. In order to get a sense of how the RSESF demonstrates the best generalisation performance on OOD testing, we randomly pick up a textured mosaic to generate 2650 augmented copies of the original image and feed them to models trained for OOD evaluation. As shown in Figure 5.8, E(2)CNN

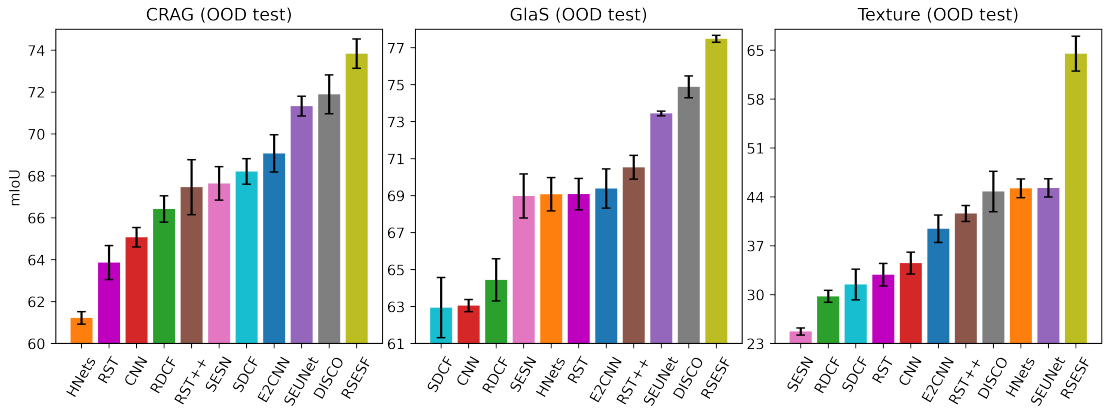


FIGURE 5.7: The OOD testing results of models on evaluated datasets. The mIoU scores (mean S.E.) of models are sorted in ascending order from left to right for visual comparison. Different colours are used to distinguish models.

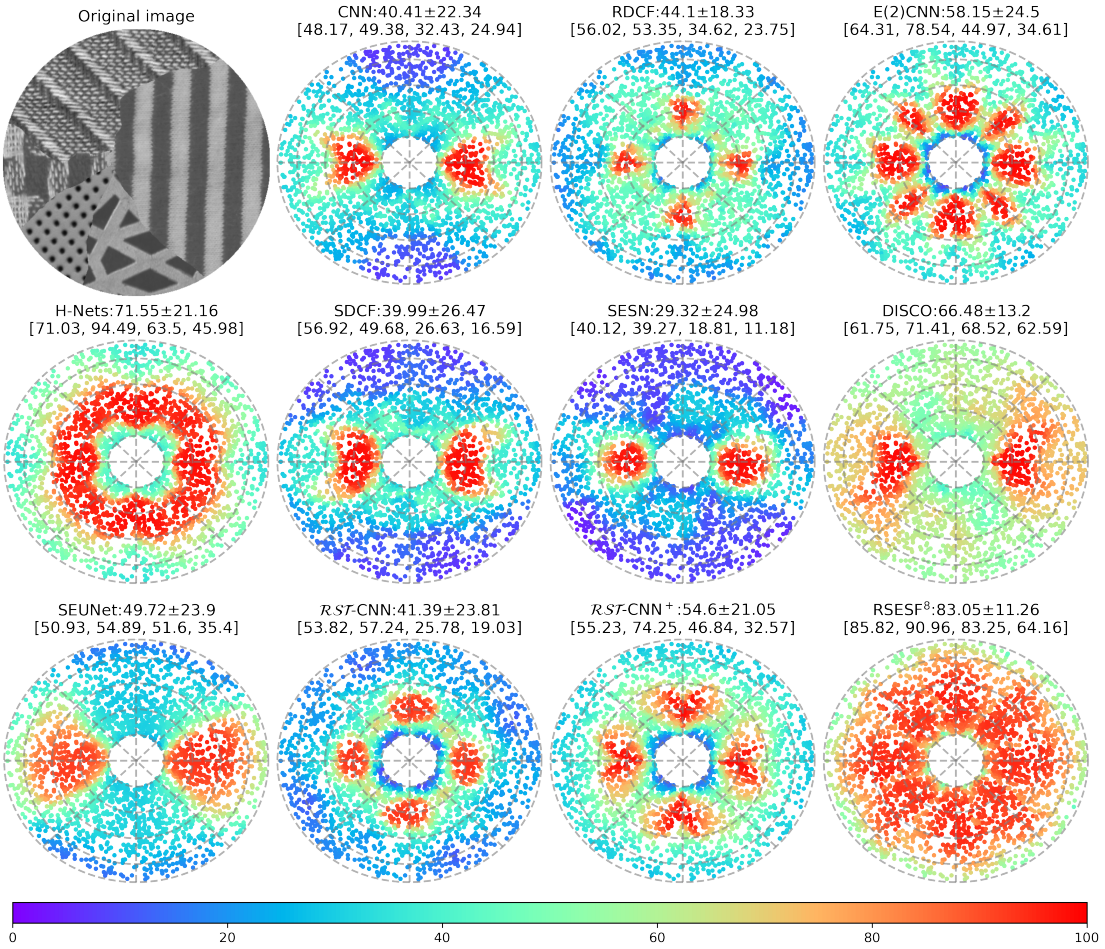


FIGURE 5.8: Polar scatter plots show mIoU scores for texture mosaics as a function of the orientation and scaling factor of an input image. 2650 texture mosaics are generated through randomly rotating (by an angle uniformly distributed on [0, 2) and re-scaling (by a factor uniformly distributed on [0.5, 2.3]) the original mosaic (shown on the top left). The radii of the five rings from inside to outside are 0.5, 1, 1.5, 2 and 2.3, which represent the corresponding scaling factors. The mIoU of each prediction is indicated with colours. The mean std of mIoU over 2650 predictions are shown on top of each plot respectively. The mean mIoU of each ring area that characterises the model's performance on different scale ranges 0.5, 1 , 1, 1.5 , 1.5, 2 , 2, 2.3 are also reported on the title of each plot.

and H-Nets show stabilised prediction to rotation variations, but they fail to generalise accurately to images that are re-scaled by factors approximately smaller than 0.75 and factors larger than 1.5. The H-Nets show more consistent prediction than E(2)CNN on images rotated by angles close to $\frac{2i \ 1}{8}, i \quad 0, \quad ,7$. This is owing to the fact that the circular harmonics is used in the filter parameterisation of H-Nets leading to full rotational equivariance, thus enabling H-Nets to capture orientation features in a fine-grained way. Although RDCF, $\mathcal{RST}$-CNN and $\mathcal{RST}$-CNN also have 8 rotation channels, their robustness is manifested only for 4 rotation angles. In terms of scale equivariant methods, for SDCF and SESN, the ability to achieve consistent segmentation on up-scaled images is limited. DISCO and SEUNet demonstrate the performance gain brought by scale equivariance, but this performance enhancement is constrained to a

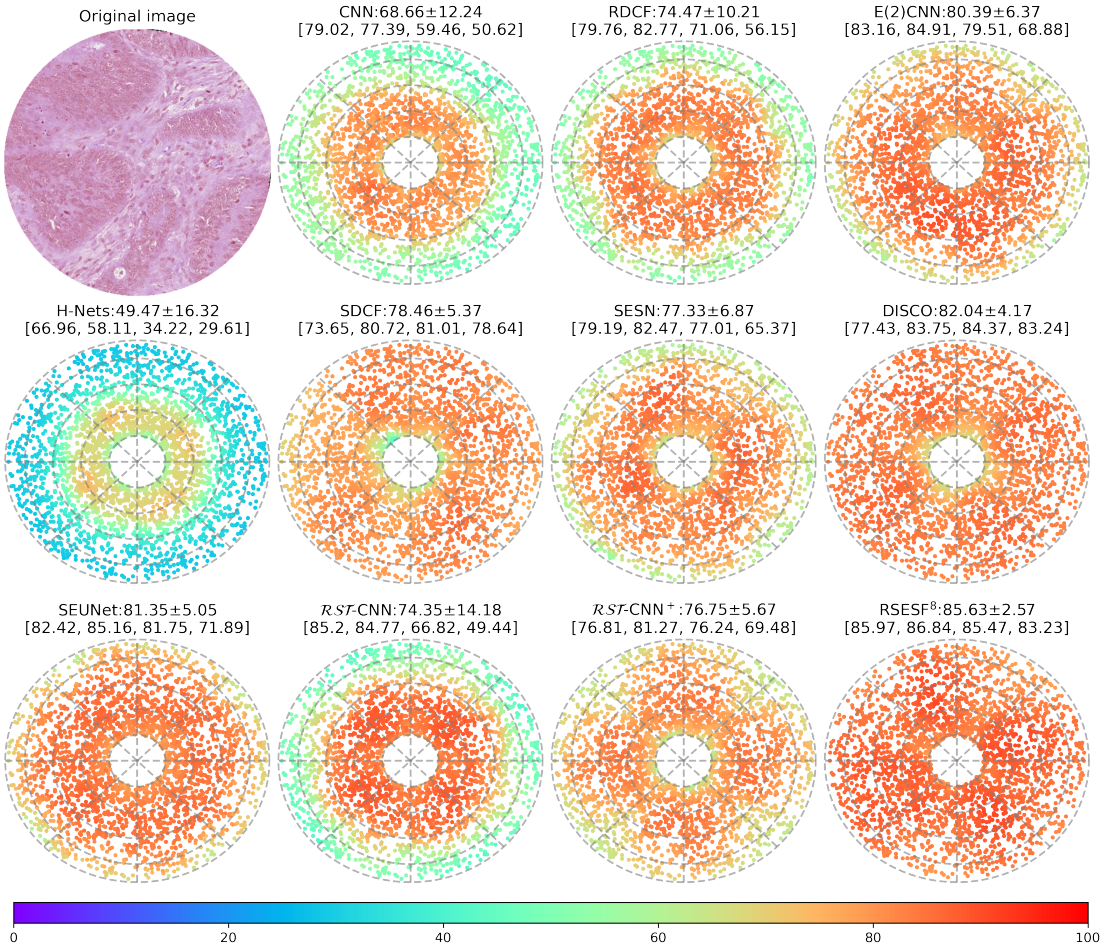


FIGURE 5.9: Polar scatter plots show mIoU scores for histopathology image as a function of the orientation and scaling factor of an input image. 2650 histopathology images are generated through randomly rotating (by an angle uniformly distributed on [0, 2) and re-scaling (by a factor uniformly distributed on [0.5, 2.3]) the original image (shown on the top left). The test image is from the CRAG dataset. The layout is the same as that of Figure 5.8.

limited range of orientation variations $\frac{}{4}, \frac{}{4}$, due to the absence of rotation equivariance. By tripling the number of filters, $\mathcal{RST}$-CNN demonstrates higher performance than $\mathcal{RST}$-CNN, but its sensitivity to orientation and scale variation is not much reduced. After extending the number of rotation channels of the SEUNet from one to eight, we construct the RSESF^8. As shown in the bottom right plot of Figure 5.8, the segmentation performance of our proposed RSESF is not severely adversely affected by variations in image orientation and scale. The RSESF maintains the performance gain brought by the scale equivariance property of the SEUNet and further boosts the performance by spreading this advantage to other orientations. However, it is worth mentioning that RSESF can only achieve discrete rotation equivariance (in our case, eight orientations). When comparing the segmentation performance of RSESF and H-Nets on a subset of images that are re-scaled by factors lie in $1, 1.5$, RSESF demonstrates inferior performance to H-Nets (90.96 *v.s.* 94.49). This highlights the superiority of continuous rotation equivariance and reveals the limitation of RSESF. Note that all of

the scatter plots show a nearly horizontal and vertical symmetry pattern, this is due to the presence of randomly horizontal and vertical flip augmentation during the training of OOD evaluation (mentioned in Figure 5.6). Some segmentation maps are presented in Appendix C.4, Figure C.5 (histopathology image) and Figure C.6 (texture mosaic).

When evaluating models' sensitivity to the orientation and scale variations on histopathology images, Figure 5.9 illustrates patterns that differ from Figure 5.8. As shown in Figure 5.9, for all methods, the orientation change of the test image does not cause a significant decrease in the segmentation performance of the model. In Figure 5.8, however, models are sensitive to orientation variation. This difference is attributed to histopathology images' inherent rotation symmetry property, i.e., features are as likely to appear in any orientation. Therefore, one can argue that models trained on histopathology images already saw patterns presented in every orientation, even though rotation augmentation is not used during training. When comparing scale equivariant methods (SDCF, SESN, DISCO, SEUNet) with conventional CNN and rotation equivariant methods (RDCF, E(2)CNN, H-Nets), scale equivariant methods demonstrate better generalisation performance than rotation equivariant ones, concerning scale variation. Surprisingly, the continuous rotation equivariant H-Nets show the worst performance on histopathology images. In the future, further analysis is needed to identify the reason for this. Among all methods compared, the proposed RSESF shows the strongest capability to resist orientation and scale variation (all of the dots are nearly red).

5.4 Ablation Study

5.4.1 Channel Squeeze at Rotation Channel

In section 5.2.4 and Figure 5.4, we mentioned that the feature maps of the last layer from each scale group are max-pooled over rotation channels to be matched with the following 1 1 convolution layer for channel squeeze and prediction generation. Here, we explore other dimension reduction methods and summarise their performance in Table 5.2. Given the feature map f_k^L that is computed by the last layer of the k^{th} scale group, the following strategies are used to transfer f_k^L C^L R H W to f_k^L C^L H W:

1. **Max-pooling f_k^L over rotation dimension.** f_k^L C^L R H W, k 1, , has components $f_{k,r}^L$, r 1, , R for each pixel, then the maximum value over rotation channels is retained, i.e., f_k^L max $\max_{1 \; r \; R} f_{k,r}^L$.

2. **Selecting a unique rotation channel R for all pixels over C^L filter channels.** We sum f_k^L along spatial and filter channel dimensions, the resultant tensor thus

TABLE 5.2: Comparison of the segmentation performance when different channel squeeze strategies are applied on the last layer. Values in the table are the mean S.E of mIoU under 5-fold cross-validation.

Channel Squeeze at the Last Layer	**Out-of-Distribution**					
	GlaS		**CRAG**		**Texture**	
Pooling over R	**78.15**	**0.15**	**74.46**	**0.51**	52.88	1.68
Selecting R over C^l	77.48	0.19	73.83	0.70	**64.49**	**2.50**
Selecting r^c for each c	77.54	0.13	73.88	0.51	54.00	0.71

reflects the overall activation magnitude of each orientation. Then the rotation channel that has the largest activation value is selected and denoted as R :

$$R \quad \arg\max_{1 \; r \; R} \sum_{x \; 1}^{H} \sum_{y \; 1}^{W} \sum_{c \; 1}^{C^L} f^L_{k,c,r,x,y} \tag{5.10}$$

then for every pixel at each filter channel we obtain $f^L_{k,c,x,y} \quad f^L_{k,c,R\ ,x,y}$.

3. **Selecting a specific rotation channel r^c_k for all pixels at each filter channel c** $1, \quad , C^L$ **.** We first sum f^L_k along spatial dimension, the resultant tensor thus reflects the overall activation magnitude of each rotation channel r and each filter channel c. Then the rotation channel with the highest mean activation values in filter channel c is selected and denoted as r^c_k:

$$r^c_k \quad \arg\max_{1 \; r \; R} \sum_{x \; 1}^{H} \sum_{y \; 1}^{W} f^L_{k,c,r,x,y} \tag{5.11}$$

then for every pixel at each filter channel c we obtain $f^L_{k,c,x,y} \quad f^L_{k,c,r^c_k,x,y}$.

As shown in Table 5.2, all three strategies yield close performance on CRAG and GlaS datasets, but the second strategy outperforms others significantly on the texture dataset. We think the reason is associated with the characteristics of datasets. Within each mask boundary, the textures of the cells and tissues appear in all orientations in the histology datasets. This is not so in texture mosaics, where each texture appears to possess only one orientation in the training set. Max-pooling over rotation channels shuffles the orientation information of textures in a mosaic thus leading to incorrect prediction. Similarly, selecting specific rotation channel r^c_k for each filter channel c may also mix up orientation information, since $r^c_k \quad r^c_k$ is not always guaranteed, where c , c denotes different filter channels. It is noted that the best performance in our model is achieved, if a unique channel R over all C^L filter channels is selected, on the texture dataset. It is therefore reasonable to assume that the reason for this best performance is that the orientation information is not mixed up among filter channels in the second strategy.

TABLE 5.3: Comparison of the segmentation performance when a different number of rotation channels is employed during inference. R 1 is not evaluated as it degenerates the RSESF to SEUNet. Values in the table are the mean S.E of mIoU under 5-fold cross-validation.

DataSet	Number of Rotation Channels (R)									
	2		4		6		8		10	
GlaS	75.18	0.09	76.78	0.14	77.31	0.20	77.48	0.19	**77.69**	0.23
CRAG	72.39	0.54	73.54	0.67	73.68	0.64	**73.83**	**0.70**	73.75	0.66
Texture	48.02	0.60	57.94	2.12	62.95	2.47	64.49	2.50	**65.17**	**2.5**

5.4.2 Flexibly Setting R While Inferencing

In section 5.2.4, we demonstrate that RSESF possesses the flexibility of being trained with only one rotation channel but it enjoys lower orientation sensitivity in inference where other rotation channels are introduced. Here we set different R values to an RSESF model that is trained with one rotation channel (no rotation and scale augmentation is performed during training) to create other 5 models, and then evaluate these models on randomly rotated and re-scaled test sets (OOD test set). The dimension reduction strategy adopted in these experiments is "Selecting R over C^l". For an RSESF model with R rotation channels, the angular spacing between adjacent channels is $\frac{2}{R}$. GPU memory usage is increased by a factor of R by raising the number of rotation channels from 1 to R. As can be seen from Table 5.3, some improvements in segmentation performance are observed in the texture dataset by raising the number of rotation channels. This is expected since texture mosaics have specific directional texture patterns. Clearly an RSESF with fewer rotation channels is not able to pick up features presented with unseen rotational angles. An inferior segmentation is therefore associated with such an RSESF model.

For the CRAG and GlaS datasets, however, due to the inherent rotational symmetry of histopathology images, the relationship between the number of rotation channels and segmentation performance is not as straightforward as it is on the texture dataset. In histopathology images, texture patterns are already presented in arbitrary orientations during training. Changing the number of rotation channels can vary a model's performance, but one can not draw a conclusion that more rotation channels lead to better performance. However, a validation set can be used to search for an optimal R.

5.5 Summary

In this chapter, we propose the RSESF, which can generalise convolutional neural networks to segment images presented in scales and orientations that do not exist in training samples. To this end, we parameterise filters by linearly combining groups of Gaussian directional derivative filters, within each one of the filters there is an additional

rotation and scale channel to guarantee rotation and scale equivariance. The scale parameters are set to be both trainable yet cover disjoint ranges. Therefore scale equivariance is achieved and specific scale preference can be found for different datasets. The rotation channel can have R filters of different orientations, spanning over 360 with an interval of $\frac{2}{R}$ to guarantee rotation equivariance. Models with RSESF filters can be trained in a memory-efficient way, as the nature of decoupled equivariant convolution gives the model flexibility of training on one orientation but inference in multiple orientations. This property overcomes the limitation of $\mathcal{RST}$-CNN [34] which is highly memory demanding. We also confirm experimentally that the RSESF achieves higher sample efficiency when compared with normal CNNs.

However, the RSESF has the following limitations that we seek to overcome in the near future:

- Although the filter constructed by using steerable basis filters can achieve continuous rotation equivariance in theory, it is intractable to have an infinite number of rotation channels in practice. This means RSESF is restricted to discrete rotations. In the future, we will explore generalising discrete rotation angles to continuous rotation angles while making the model scale equivariant.
- Currently, we squeeze feature maps from N_r channels to channels at the penultimate layer and then use the strategy proposed in section 4.2.5 for prediction generation. However, other aggregation methods that directly work on N_r dimensions may pick up the best orientation and scale channels for test images, to lead to better segmentation.
- The proposed framework is designed for global rotation-scale equivariance, i.e., assuming that multiple objects are all in the same scale and orientation, in each single image. Therefore, the adopted strategy that selects a specific scale and rotation channel for each test image works for this scenario. This, however, ignores cases where multiple objects may show up in different orientations and scales, in a single image. Therefore, it would be interesting to extend the RSESF from a global equivariant to a local equivariant framework.

Chapter 6

Conclusions and Future Work

This dissertation investigates the problem of segmentation of histopathology images, using fully convolutional-type neural networks. Several algorithms or frameworks are proposed to enhance the accuracy of segmentation maps from different perspectives and address the challenge of magnification level variations. In this chapter, we first summarise the findings and limitations identified in the previous chapters and then we discuss potential future directions.

6.1 Conclusions

In chapter 3, we first revisit the UNet^e and UNet++ architectures and then point out the limitations of existing frameworks. Firstly, embedded training of the model tends to correlate learnt features. Such a training scheme would conflict with the nature of ensemble methods, where features are supposed to be de-correlated. Secondly, simply averaging the output of constituent models ignores the performance of each individual UNet within the ensemble architecture and thus results in limited or even worse segmentation performance. To suppress the negative effects introduced by these causes, we propose a stage-wise additive training algorithm named as ADS_UNet, wherein constituent UNets with different depths are trained in an isolated fashion on re-weighted samples for the purpose of forcing them to focus on previously misclassified samples. By doing so, we experimentally validate that the diversity of features is increased from the indication of decreased CKA similarity of features. After being trained, we aggregate predictions from base UNets and generate the final prediction by weighted combination, where the weighting factors reflect the performance of base UNets. The effectiveness of the AdaBoost-based ensemble is experimentally validated on four histopathology datasets. Apart from achieving better segmentation performance, we arrive at a more computational and memory-efficient architecture by

modifying the deep supervision scheme. The limitation of ADS_UNet is that the sample re-weighting criteria is in a coarse granularity that can not highlight regions that are difficult to distinguish.

Chapter 4 addresses the problem of generalising CNNs to segmentation tasks on histopathology images of unknown magnification-level. The lack of magnification information on histopathology snapshots greatly hinders the quantitative analyses needed for such snapshots. In chapter 4, we impose scale equivariance to empower CNNs with the capability of resisting scale variations, to achieve more stable segmentation. We parameterised multi-scale convolutional filters with linear combinations of Gaussian derivative filter basis elements. By arranging scale parameters of filters to span disjoint ranges and being tuned during training, scale equivariance is guaranteed layer by layer and the optimal setting of scale parameters for the dataset is decided by the model itself. The work done in chapter 4 suggests that encoding scale equivariance into CNNs is beneficial for models' generalisation capability when scale variations exist between datasets. The limitation of the proposed scale equivariant UNet is that the optimal prediction selection/ensemble strategy that could make the best-fit scale branch contribute the most remains unexplored.

In chapter 5, we extend the scale equivariant UNet proposed in chapter 4 to be a joint rotation-scale equivariant model. The motivation behind this extension is to utilize the inherent rotation symmetry of histopathology images as prior knowledge, such that CNNs do not need to devote a large proportion of parameters to explicitly learn feature patterns that correspond to different orientations. We propose RSESF, which adopts Gaussian directional derivatives as a basis filter to parameterise groups of multi-scale multi-orientation convolutional filters. We conduct a thorough comparative analysis of various rotation and/or scale equivariant CNNs and demonstrate that the RSESF achieves state-of-the-art generalisation performance on the task of gland segmentation and texture segmentation with a fraction of the parameter budget of conventional CNNs and recent top-performing equivariant models. Apart from performance improvements, the RSESF can be trained in a memory-efficient way and possesses the flexibility of adjusting orientation sensitivity by setting different numbers of rotation channels at inference time. The main limitation of the RSESF is that it can only achieve discrete rotation-scale equivariance, which limits its generalisation capability to a finite number of orientations and scales.

6.2 Future work

In the summary sections of chapter 3, 4 and 5, we identify the limitation of our proposed methods and point out the directions that can be explored to eliminate shortcomings of the methods proposed here. Apart from the future work mentioned previously,

further ideas for future research are highlighted as follows.

6.2.1 Slide-Level Segmentation

WSIs pose unique challenges when training deep learning models. The characteristic of large size makes it necessary to break each WSI down into small-sized patches for analysis. In this thesis, all of the proposed algorithms or frameworks primarily focus on patch-level image segmentation, ignoring the global context correlation of neighbouring patches. However, a slide-level segmentation is more important as it provides the pathologists with more comprehensive and intuitive information about the severity of the disease. Stitching segmented patches together is the most straightforward way to form slide-level segmentation, but it may suffer from inconsistent segmentation at patch boundaries. Modelling the spatial adjacency between patches could be beneficial for achieving smooth segmentation, from a global perspective. Some early attempts have shown that modelling spatial correlations by conditional random fields [67] or graph neural networks [65, 93] obtains probability maps of patch predictions with better visual quality. These methods, however, assign a unique label to each patch ignoring the cases where a class boundary goes through the patch. Therefore, their slide-level segmentation is in coarse grain. This drawback is especially obvious when zooming into local regions where boundaries exist. In the future, it would be interesting to explore methods that perform fine-grained and spatially correlated slide-level segmentation.

6.2.2 Automatic Generation of Diagnostic Report

In a clinical pathology diagnosis workflow, the key step is that pathologists examine the biopsy slice by either tuning the focal length of the microscope or scrolling the wheel of the mouse to zoom in or zoom out. During this procedure, some regions of interest that carry rich diagnostic information will be saved as snapshots for further investigations. Following this, pathologists will carefully examine those snapshots and then summarise findings to form a diagnostic report, with representative snapshots attached. Currently, most of the efforts in the research of computational pathology are put on the examination part, i.e., building AI algorithms to extract features that are descriptive enough to distinguish healthy and diseased tissues. Although the ultimate goal of algorithms is to achieve an automatic diagnosis, current works just classify tissues into healthy and diseased groups, without providing the text interpretation of the diagnosis. Therefore, a computational pathology platform that completes the entire workflow done by pathologists should also include the automatic generation of diagnostic reports. This is not just for the integrity of computational pathology algorithms, but also for improving their interpretability, and making them more trustworthy.

Currently, the research on diagnostic report generation mainly relies on existing algorithms and frameworks from natural image caption generation, which is the fusion of natural language processing and computer vision techniques. However, histopathology images have some specific properties such as inherent rotation symmetry that do not exist in natural images. Directly transferring existing algorithms from the natural image domain to the histopathology images does not fully utilise this prior knowledge and thus may lead to limited outcomes. Moreover, existing datasets for histopathology image caption generation consist of snapshots and image patches with mixed or unknown magnification levels. This large magnification variation is another bottleneck that makes the task challenging. In the future, we wish to apply our proposed rotation-scale equivariant method RSESF to the feature encoding part of diagnostic report generation frameworks. We expect that this joint rotation and scale equivariance of RSESF will extract more descriptive features than conventional CNN, under the domain of histopathology, and will thus lead to more accurate caption generation.

Appendix A

Enhancing Segmentation via Ensemble Learning

A.1 Python code for counting incorrect labels in down-scaled masks

Here we provide the detailed Python code for quantifying incorrect labelling information existing in down-sampled masks.

Appendix B

Scale Equivariance for Robust Segmentation

B.1 Visualising the Scale Equivariance Error

To demonstrate the effectiveness of lowering equivariance error by convolving images with multi-scale filters, we convolve images at different scales with 5 filters, paring feature maps and then calculate the equivariance error as:

$$_{s,k,k} \quad \frac{1}{N}\,{}_{i\ 1}^{N} \frac{S_s\ F_k\ f_i \quad F_k\ S_s\ f_i\ {}_2^2}{S_s\ F_k\ f_i\ {}_2^2},\ s \quad ,\ k,k \quad 1, \quad , \tag{B.1}$$

where f_i is an image, F_k and F_k are filters with scale parameters $_k$ and $_k$, S_s is a scaling operation with factor s. Thus, given filters and two images at different scales, we arrive at a equivariance error matrix. Where each element represents the equivariance error between feature maps, which are obtained by convolving images of different scales with different filters. As shown in Figure B.1, for the feature map pair that produces the maximal matching, the ratio of scales between images is equal (or close) to the ratio of $_k$s between filters. For example, in Figure B.1p, the ratio between s and the ratio between image scales is the same ($\frac{2}{0.5}$ $\frac{4}{1}$). The same phenomenon can be observed from images re-scaled by factors of 0.5 and 2 (Figure B.1e and B.1l). For images whose scales are not divisible, the matching degree between feature maps obtained by convolving the filter with the ratio closest to the image ratio is the highest. For example, in Figure B.1f, the ratio between images ($\frac{1}{0.59}$ 1.69) is close to the ratio between s ($\frac{2.5}{1.5}$ 1.67). Thus, we experimentally validated that the scale equivarance error can be reduced by convolving images at different scales with appropriate filters whose scale is corresponded to the scale of the images.

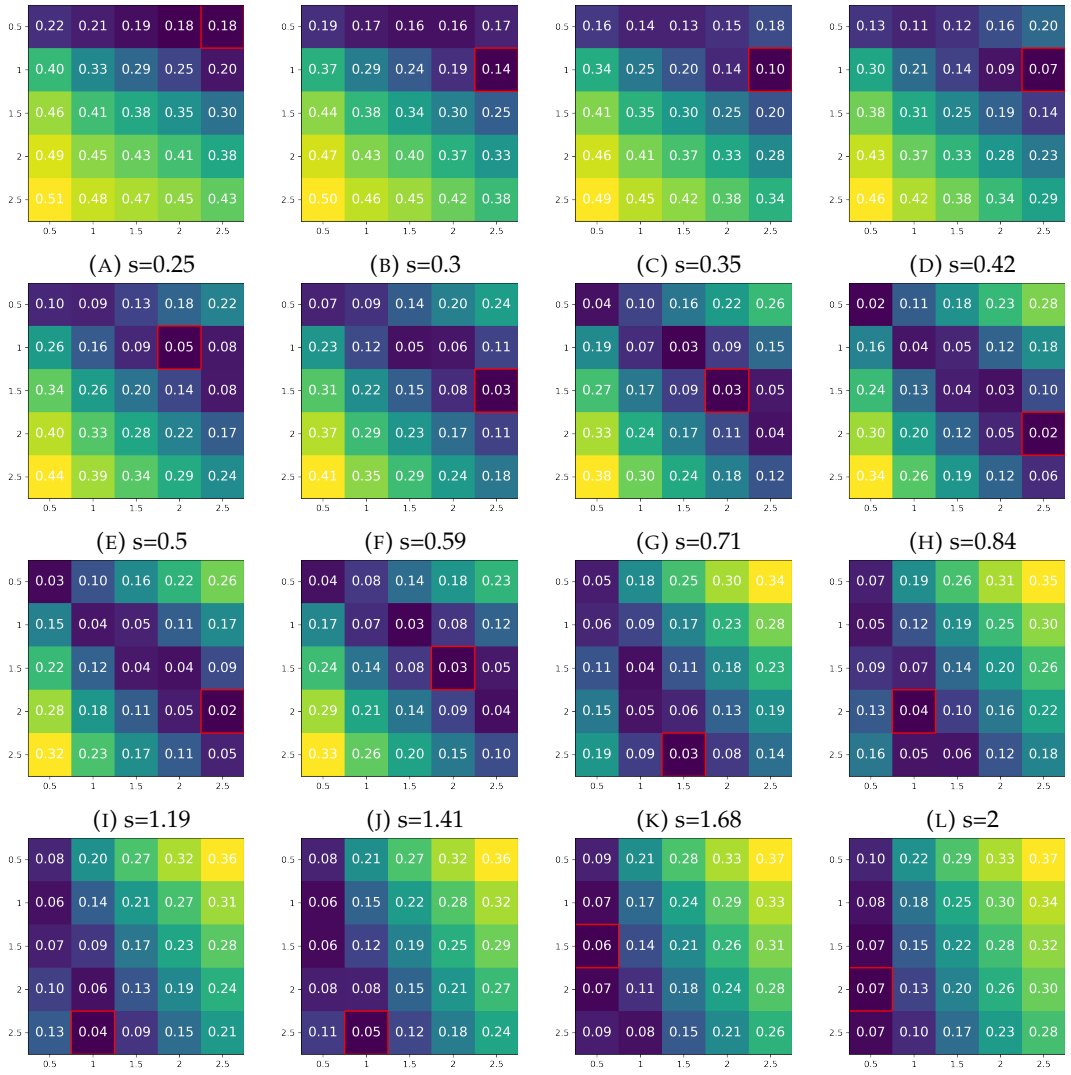


FIGURE B.1: Scale equivariance error of feature maps. Each plot shows the equivariance errors between feature maps of the original image and the re-scaled image. In the title of each plot, s denotes the scale factor. The x-axis and the y-axis of each plot denote the of the filter that is used to convolve with the original image and the re-scaled image, respectively. In each plot, the number on the grid denotes the equivariance error between feature maps S_s F_k f and F_k S_s f . The lowest equivariance error in each 5 5 error matrix is highlighted by a red box.

B.2 Visualisation of Model Prediction

To better understand the SEUNet, we visualise segmentation maps generated by the SEUNet and other compared models on input images at different scales. As shown in Figure B.2 and B.3, the SEUNet can retain a relatively decent prediction when compared with other methods.

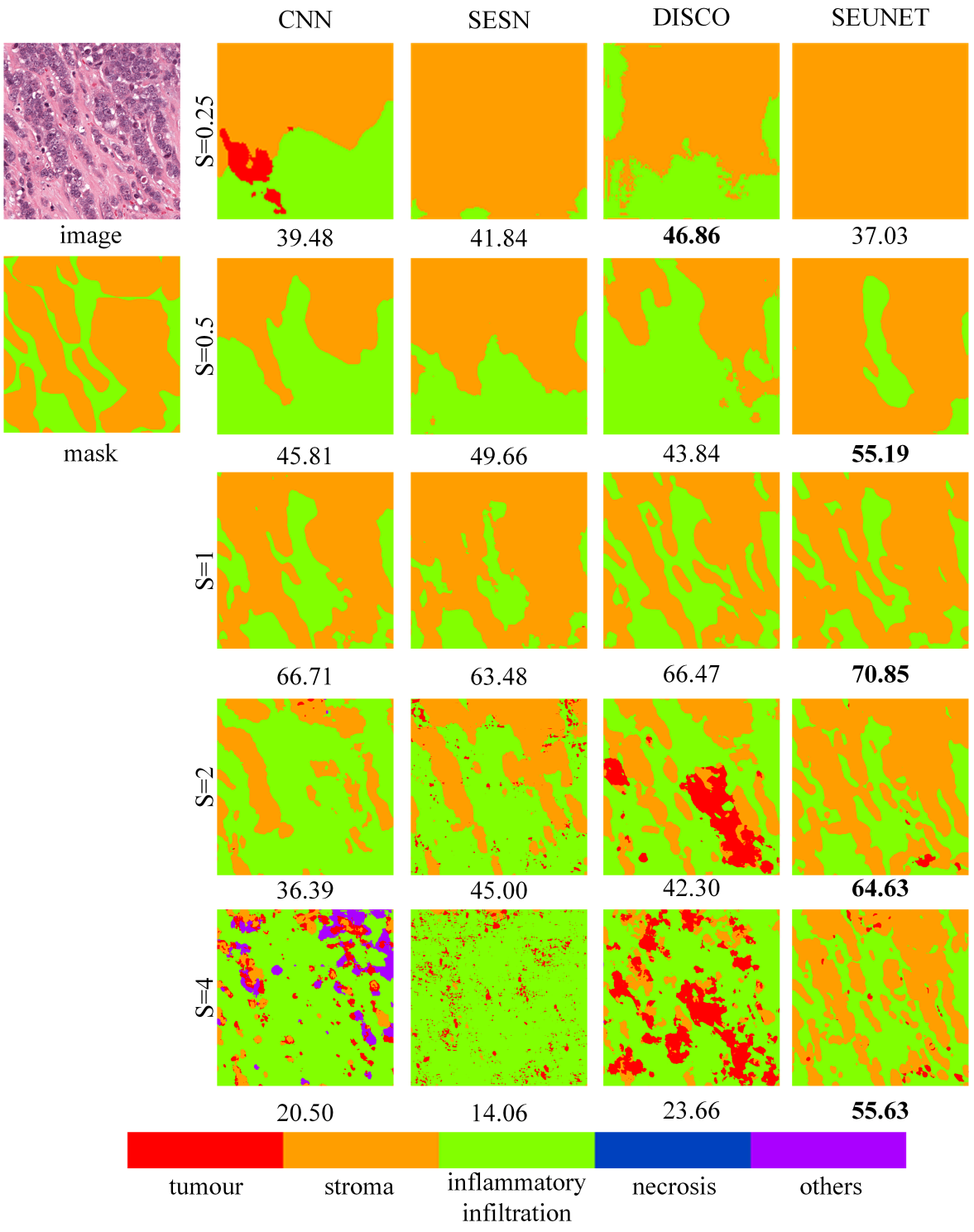


FIGURE B.2: Visual comparison on the BCSS dataset. The mIoU score of each prediction is reported below the segmentation map. The highest score is highlighted in bold. Each column shows segmentation maps of a model on an image re-scaled by different scaling factors (S 1 denotes the original scale).

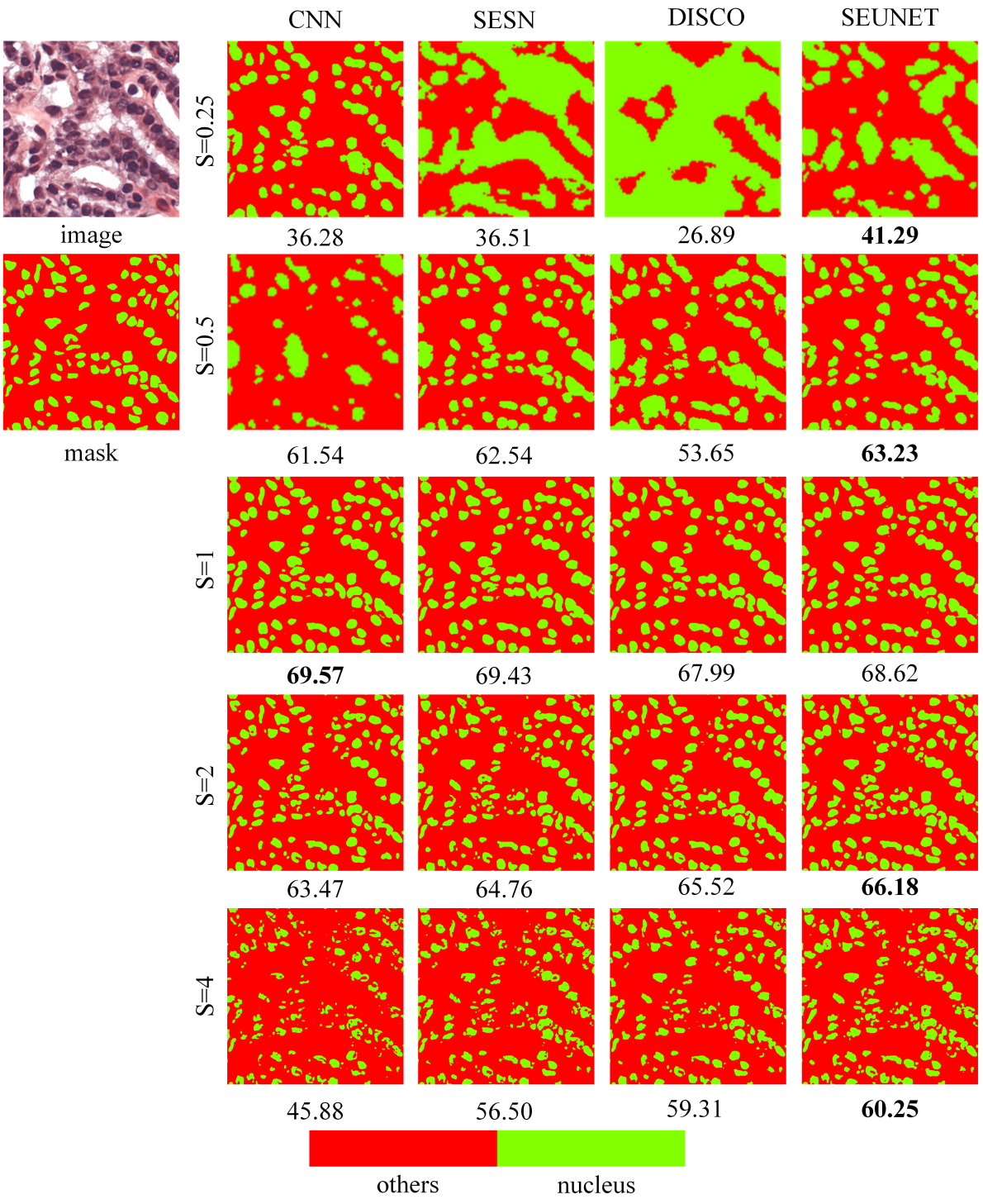


FIGURE B.3: Visual comparison on the MoNuSeg dataset. The IoU score of each prediction is reported below the segmentation map. The highest score is highlighted in bold. Each column shows segmentation maps of a model on an image re-scaled by different scaling factors (S 1 denotes the original scale).

Appendix C

Joint Rotation-Scale Equivariance

C.1 The Second Order Directional Filter Basis

Here we derive the filter basis of order 2, as the highest order of the Gaussian derivative used in our experiments is 2. Denoting $z \quad G\ x,y; \quad$ and $z \quad G^1\ x,y; \quad$, then the second order directional derivative of z with respect to angle $\quad$ can be calculated by:

$$
\begin{aligned}
G^2\ x,y; \qquad & \cos\ ,\ \sin \qquad \frac{z}{x}, \frac{z}{y} \\
& \cos\ \frac{z}{x} \quad \sin\ \frac{z}{y} \\
& \cos\ \frac{G^1\ x,y;}{x} \quad \sin\ \frac{G^1\ x,y;}{y} \\
& \cos\ \cos\ \frac{^2G\ x,y;}{x^2} \quad \sin\ \frac{^2G\ x,y;}{x\ y} \\
& \sin\ \cos\ \frac{^2G\ x,y;}{y\ x} \quad \sin\ \frac{^2G\ x,y;}{y^2} \\
& \cos^2\ \frac{^2G\ x,y;}{x^2} \quad 2\cos\ \sin\ \frac{^2G\ x,y;}{y\ x} \quad \sin^2\ \frac{^2G\ x,y;}{y^2} \\
& \cos\ _x\ \sin\ _y\ ^2G\ x,y;
\end{aligned}
\tag{C.1}
$$

C.2 Sample Efficiency Analysis

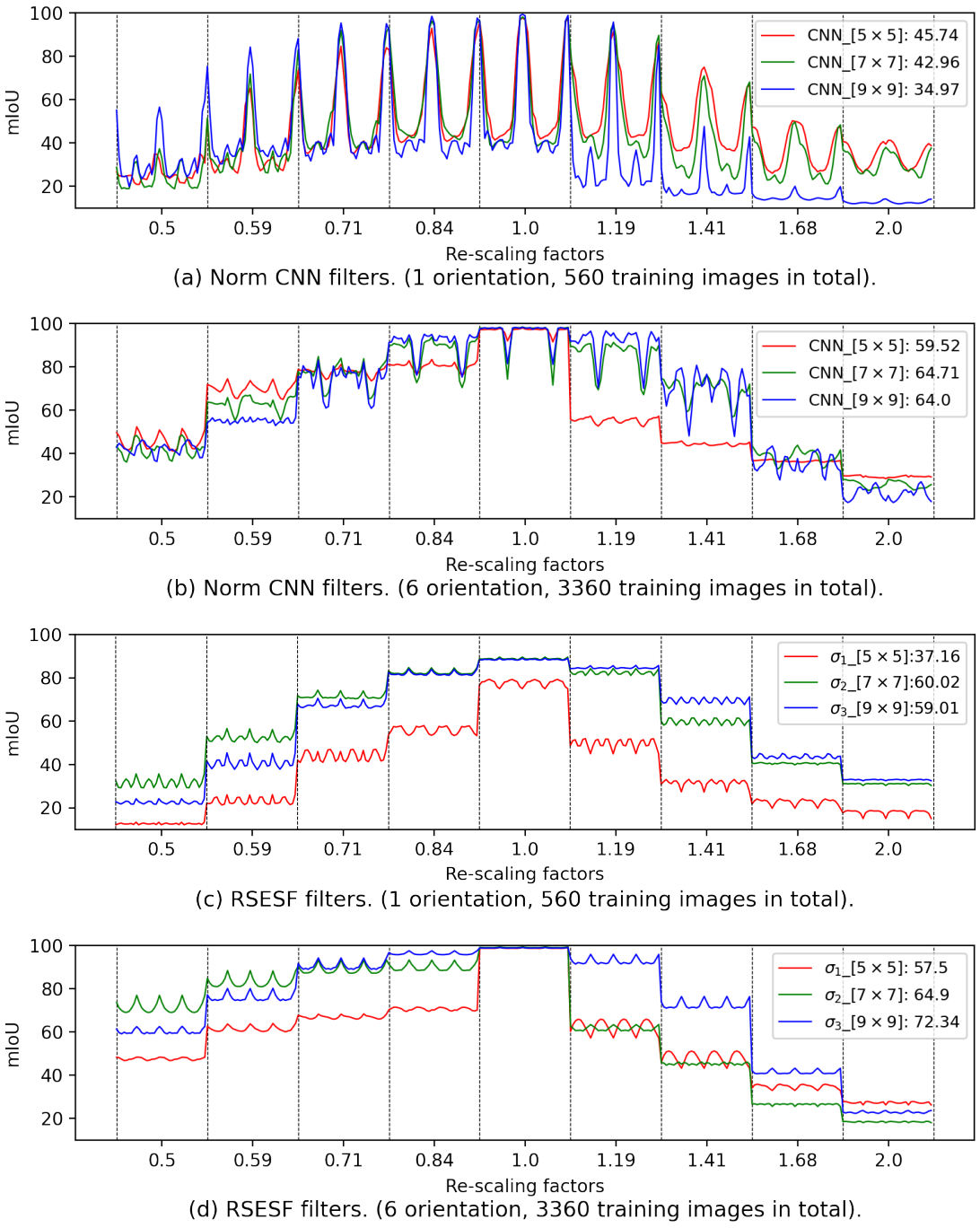
Apart from experiments conducted in section 5.3, here we design experiments in which UNets with conventional CNN filter and RSESF are trained either with or without rotation augmentation to demonstrate the superiority of RSESF on sample efficiency.

Dataset configuration. We use the Texture dataset for model training and evaluation.

- Training set 1, in which 80% of texture mosaics are presented in one particular orientation, is used for model training under the setting without rotation augmentation.
- Training set 2 is the rotation-augmented version of training set 1, where each texture mosaic is presented in 6 angles (0 , 60 , 120 , 180 , 240 , 300). Therefore there are 3456 texture mosaics in total. The training set 2 is used for model training under the setting of rotation augmentation.
- Test set. For evaluation, we re-scale and rotate the texture mosaics of the original texture test set by 36 angles $\frac{2}{36}\ \frac{r}{r}\ \frac{36}{1}$ and 9 scaling factors $\frac{\sqrt[4]{2}^{s}}{2}\ \frac{8}{s}\ _{0}$. Therefore there are 324 subsets of mosaics, consisting of 46656 144 9 36 texture mosaics in total.

Model settings. We create an UNet with RSESF filters that have 3 scale groups. By constraint the upper and lower bounds of $\frac{l}{k}$ using equation (5.8), we therefore constraint the size of filters at each group to be 5 5 , 7 7 and 9 9 , for every layer. For a fair comparison, we match the size of norm CNN filters with that of each scale group of RSESF filters. In detail, we create CNN_ 5 5 , CNN_ 7 7 and CNN_ 9 9 , where CNN_ k k means that the size of filter is set to be k k for every layer of the UNet. The number of filters is set to be 60, 120, 240, 480 and 960, at each depth of the CNN-based UNet. For RSESF-based UNet, the number of filter channels is divided by 3. When evaluating, we generate a segmentation map for each scale group and report their performance separately.

Results. As shown in Figure C.1, when models are trained without rotation augmentation, CNNs only show competitive performance on mosaics that are presented at the same orientation as training mosaics and a large fluctuation can be seen from Figure C.1(a). In contrast, as shown in Figure C.1(c), RSESF demonstrates relatively stable prediction over orientations. When models are trained with rotation augmentation (with 5 times more training samples), both the performance and robustness of CNNs are greatly improved, while some fluctuations between orientations still remain. The overall performance of RSESF-based UNet also benefits from rotation augmentation. It is worth mentioning that RSESF-based UNet trained on 560 mosaics achieves close performance to CNN-based UNet trained on 3360 mosaics (CNN_ 7 7 vs. $_{2\_}$ 7 7 and CNN_ 9 9 vs. $_{3\_}$ 9 9). In addition, the number of filter channels of RSESF is only one-third of that of CNN. This comparison highlights the higher sample efficiency of RSESF.



(a) Norm CNN filters. (1 orientation, 560 training images in total).

(b) Norm CNN filters. (6 orientation, 3360 training images in total).

(c) RSESF filters. (1 orientation, 560 training images in total).

(d) RSESF filters. (6 orientation, 3360 training images in total).

FIGURE C.1: mIoU score measured on multi-scale multi-orientation texture mosaics. Within each column (separated by vertical dashed lines), images are re-scaled by the same factor shown in the x-axis and rotated by 36 different angles (no show, 0 centred in each column). The averaged mIoU over all orientations and scales are reported on legends.

C.3 Visualisation of Equivariance Error

We convolve images that are either rotated, re-scaled, or simultaneously rotated and re-scaled with RSESF filters possessing 4 rotation and 5 scale channels, pairing their

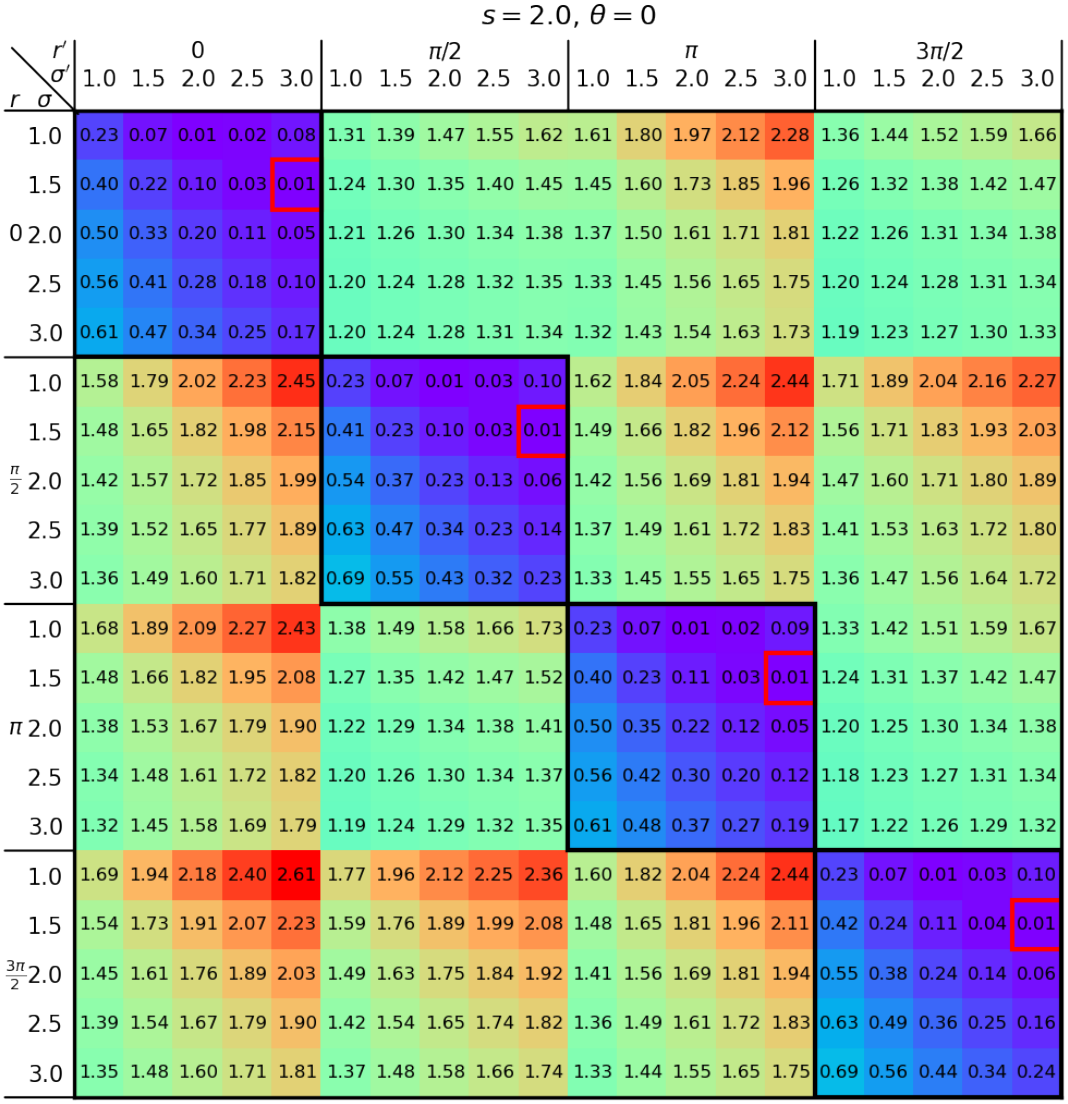


FIGURE C.2: Equivariance error of feature maps between the original image and the image up-scaled by a factor of 2. The y-axis and x-axis denote the angle r, r and the scale parameter , of filters $F_{r,}$ and $F_{r,}$. Inside each 5 5 block, filters are in the same orientation but vary in scale. For each angle r of f , the best block of filters with angle r is highlighted by black borders. Inside the best-matched group of filters, the best-matched scale of filters is also highlighted in red borders. Here, the 'best match' denotes the lowest equivariance error.

feature maps with that of the original image and then calculate the equivariance error as:

$$_{s,k,k} \quad \frac{1}{N} \sum_{i \ 1}^{N} \frac{T_{,s} F_{r,} \quad f_i \quad F_{r,} \quad T_{,s} f_i \quad \frac{2}{2}}{T_{,s} F_{r,} \quad f_i \quad \frac{2}{2}}, \ s, \ , \quad , \ ,r,r \quad 0,2 \quad . \tag{C.2}$$

where f_i is an image, $F_{r,}$ and $F_{r,}$ are filters with orientation and scale parameters $r,$ and $r,$, $T_{,s}$ is a rotation and scaling transformation with angle and scaling factor s. Thus, given RSESF filters with 4 rotation and 5 scale channels and two images

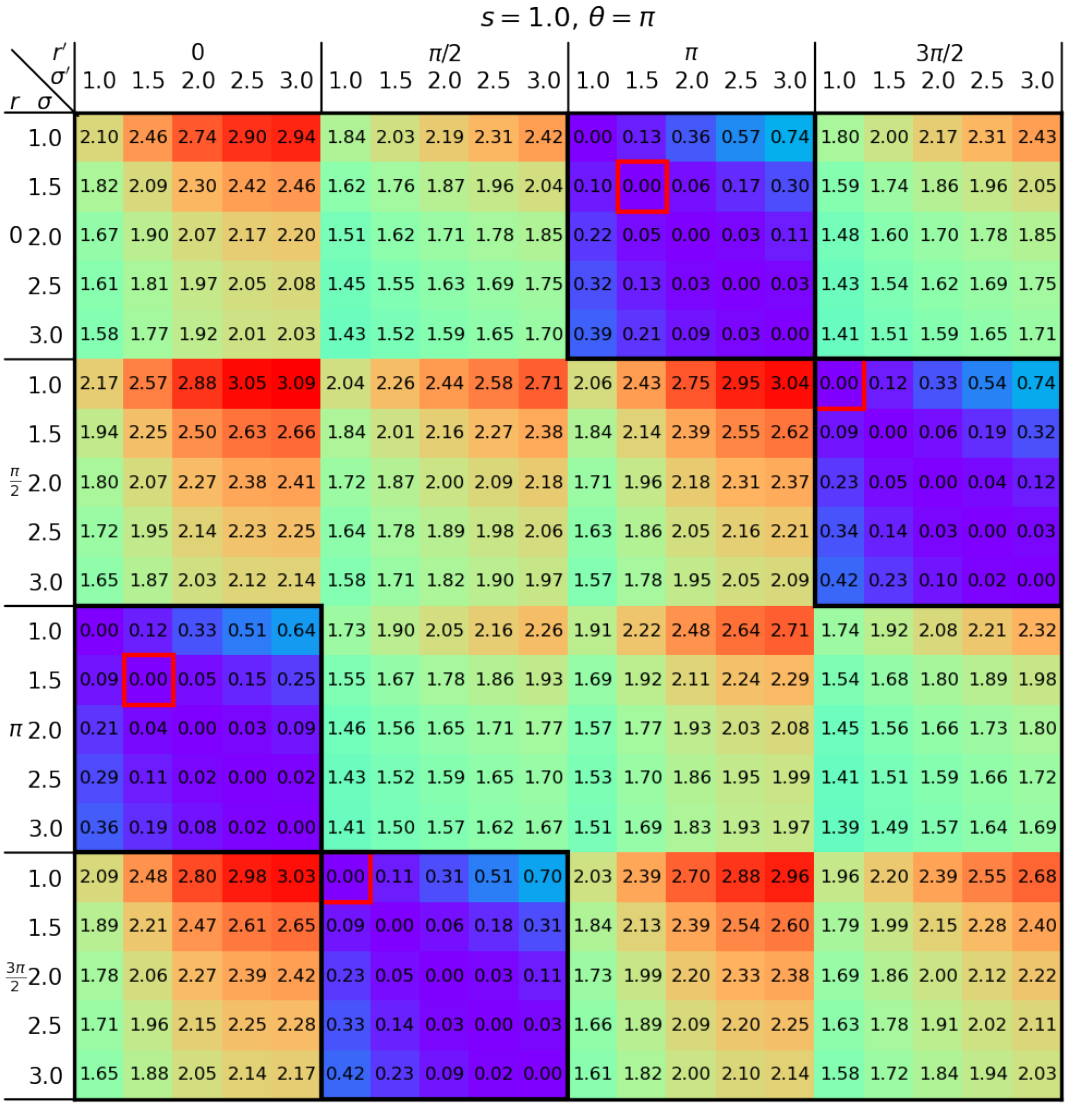


FIGURE C.3: Equivariance error of feature maps between the original image and the image rotated by an angle of .

at different scales and orientations, we arrive at a 20 20 equivariance error matrix. Each element represents the equivariance error between feature maps, which are obtained by convolving images with different filters. Different settings of and s are used to evaluate the equivariance capability of RSESF with regard to rotation, scaling, and joint rotation-scaling transformation:

- **Measuring scale equivariance.** $0, s$ 2 . Results shown in Figure C.2
- **Measuring rotation equivariance.** $, s$ 1 . Results shown in Figure C.3
- **Measuring joint rotation-scale equivariance.** $, s$ 2 . Results shown in Figure C.4

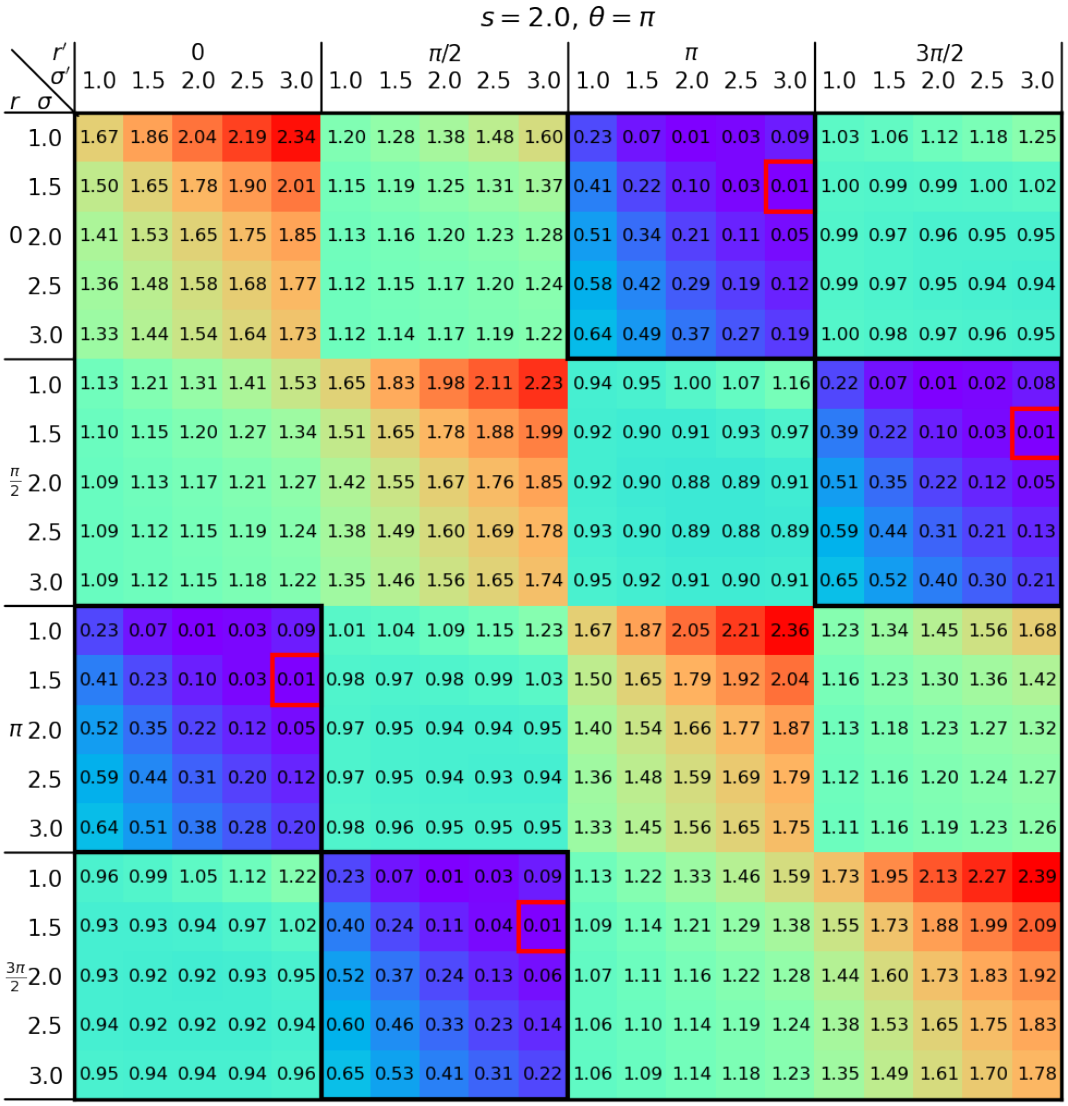


FIGURE C.4: Equivariance error of feature maps between the original image and the image rotated by an angle of and up-scaled by a factor of 2.

As shown in Figure C.2, for the feature map pair that produces the maximal matching (minimal equivariance error), the ratio of scales between images is equal to the ratio of and between filters. For example, the ratio between and and the ratio between image scales is the same ($\frac{3.0}{1.5}$ 2). This observation is consistent with that of Appendix B.1.

As shown in Figure C.3, differing from Figure C.2, rotating the image results in the shift of best-matched angle r (shifted from r r to r r). Since the transformed image is just rotated but remains in the original scale. The ratio of best-matched scales of filters is 1, within each highlighted 5 5 block.

Figure C.2 and C.3 measure the scale and rotation equivariance of the RSESF filters, separately. Figure C.4 measures the equivariance of the RSESF filter under the joint

rotation-scale transformation. As seen in Figure C.4, rotating and re-scaling the image result in both the shift of best-matched angle r (consistent with Figure C.3) and the shift of best-matched scale parameter (consistent with Figure C.2).

C.4 Prediction visualisation

In Figure C.5, we select a randomly rotated and re-scaled image from the test set of CRAG for visualisation. Some segmentation maps of texture mosaics that are presented in different scales and/or orientations are shown in Figure C.6.

Figure C.6 provides some visual clues of how different type of models demonstrate their superiority on different versions of test images. As seen from these two figures, when the test mosaic is rotated by 58 (without re-scaling it), E(2)CNN, H-Nets and RSESF can maintain relatively high prediction accuracy (80.90 v.s. 80.22 v.s. 74.82). The reason that H-Nets outperforms RSESF is that H-Nets has continuous rotation equivariance property while the orientation of 58 is not encoded in RSESF. When the test mosaic is re-scaled (without rotating it), all of the scale equivariant models, SDCF (82.46), SESN (82.44) and SEUNet (92.85) show more accurate segmentation. When the test mosaic is rotated and re-scaled simultaneously, although the performance of all models degrades, the RSESF is the one that demonstrates the best performance.

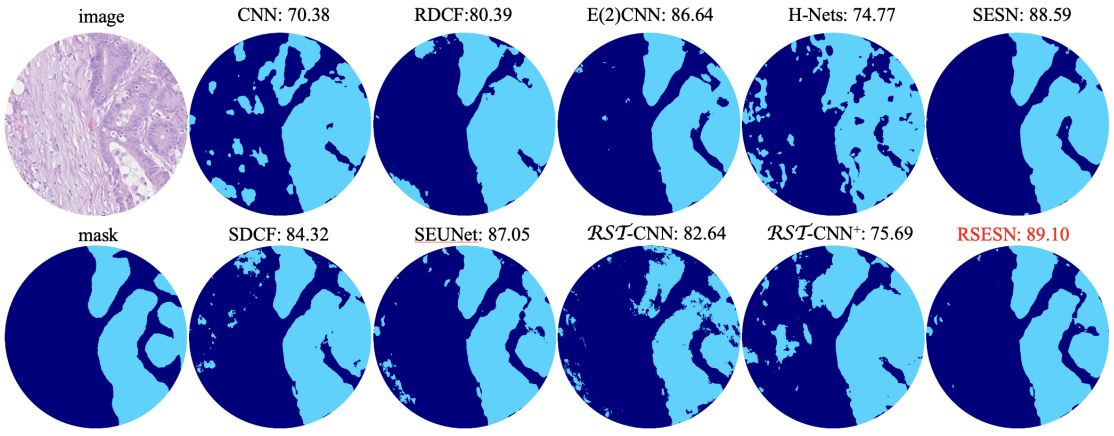


FIGURE C.5: Images, masks, and predictions generated by models. The mIoU score of each prediction is shown on top of the segmentation map. The rotation angle and scaling factor of the selected histopathology image are r 184 , s 1.97 . Light blue and dark blue represent gland and non-gland, respectively.

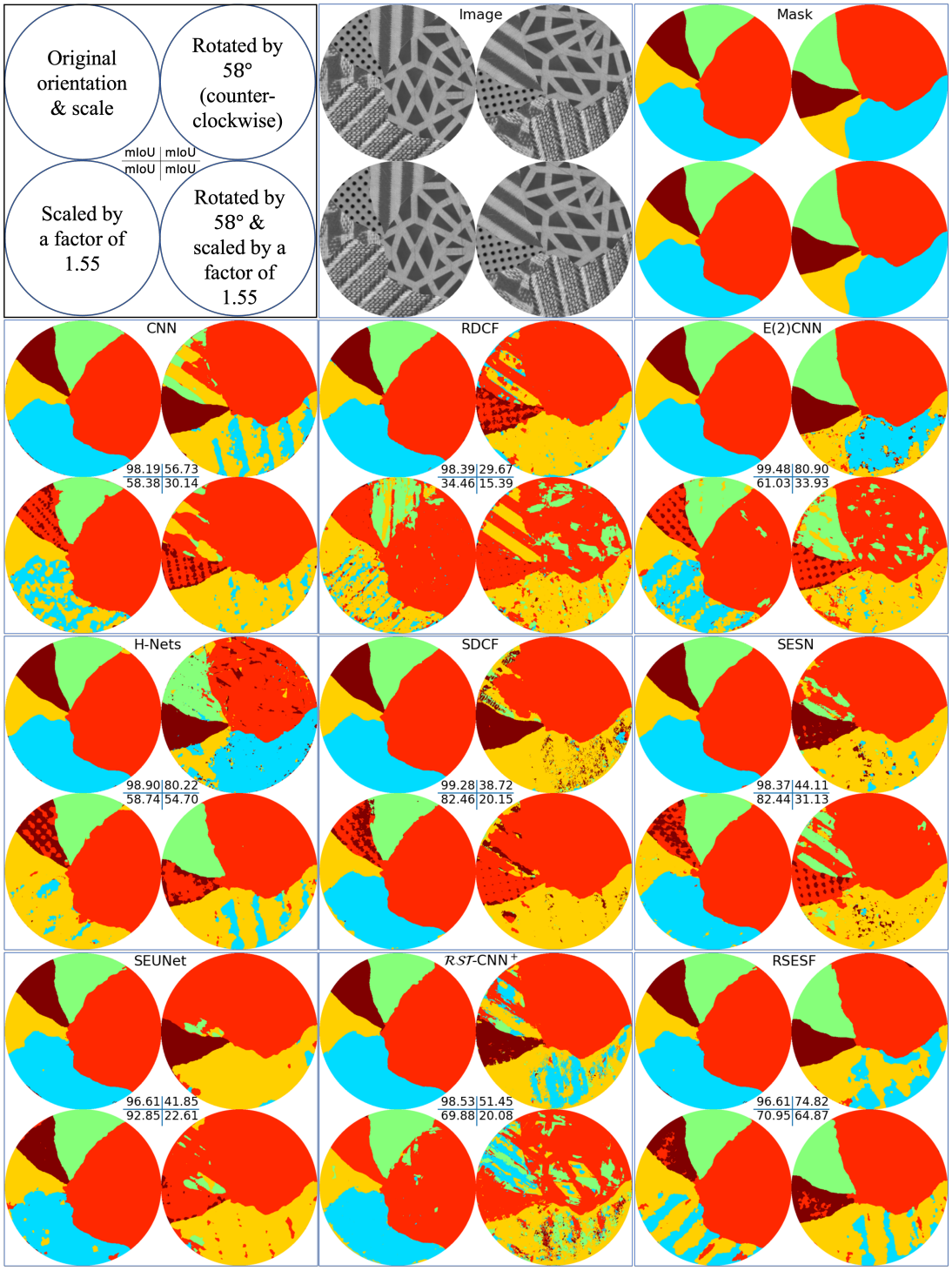


FIGURE C.6: Visualisation of segmentation maps and mIoU scores, where the test image is either rotated, re-scaled, or jointly rotated and re-scaled. The mIoU score is reported below each prediction.

References

[1] Esther Abels, Liron Pantanowitz, Famke Aeffner, Mark D Zarella, Jeroen van der Laak, Marilyn M Bui, Venkata NP Vemuri, Anil V Parwani, Jeff Gibbs, Emmanuel Agosto-Arroyo, et al. Computational pathology definitions, best practices, and recommendations for regulatory guidance: a white paper from the digital pathology association. *The Journal of pathology*, 249(3):286–294, 2019.

[2] Mohamed Amgad, Habiba Elfandy, Hagar Hussein, Lamees A Atteya, Mai AT Elsebaie, Lamia S Abo Elnasr, Rokia A Sakr, Hazem SE Salem, Ahmed F Ismail, Anas M Saad, et al. Structured crowdsourcing enables convolutional segmentation of histology images. *Bioinformatics*, 35(18):3461–3467, 2019.

[3] Ruqayya Awan, Korsuk Sirinukunwattana, David Epstein, Samuel Jefferyes, Uvais Qidwai, Zia Aftab, Imaad Mujeeb, David Snead, and Nasir Rajpoot. Glandular morphometrics for objective grading of colorectal adenocarcinoma histology images. *Scientific reports*, 7(1):1–12, 2017.

[4] Aharon Azulay and Yair Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? *arXiv preprint arXiv:1805.12177*, 2018.

[5] Peter Bankhead, Maurice B Loughrey, José A Fernández, Yvonne Dombrowski, Darragh G McArt, Philip D Dunne, Stephen McQuaid, Ronan T Gray, Liam J Murray, Helen G Coleman, et al. Qupath: Open source software for digital pathology image analysis. *Scientific reports*, 7(1):1–7, 2017.

[6] Neslihan Bayramoglu, Juho Kannala, and Janne Heikkilä. Deep learning for magnification independent breast cancer histopathology image classification. In *2016 23rd International conference on pattern recognition (ICPR)*, pages 2440–2445. IEEE, 2016.

[7] Babak Ehteshami Bejnordi, Mitko Veta, Paul Johannes Van Diest, Bram Van Ginneken, Nico Karssemeijer, Geert Litjens, Jeroen AWM Van Der Laak, Meyke Hermsen, Quirine F Manson, Maschenka Balkenhol, et al. Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. *Jama*, 318(22):2199–2210, 2017.

[8] Erik J Bekkers. B-spline cnns on lie groups. In *International Conference on Learning Representations*, 2019.

[9] Erik J Bekkers, Maxime W Lafarge, Mitko Veta, Koen AJ Eppenhof, Josien PW Pluim, and Remco Duits. Roto-translation covariant convolutional networks for medical image analysis. In *International conference on medical image computing and computer-assisted intervention*, pages 440–448. Springer, 2018.

[10] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19, 2006.

[11] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[12] M Alvaro Berbís, David S McClintock, Andrey Bychkov, Jeroen Van der Laak, Liron Pantanowitz, Jochen K Lennerz, Jerome Y Cheng, Brett Delahunt, Lars Egevad, Catarina Eloy, et al. Computational pathology in 2030: a delphi study forecasting the role of ai in pathology within the next decade. *EBioMedicine*, 88, 2023.

[13] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[14] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[15] Jun Cheng, Xiaokui Mo, Xusheng Wang, Anil Parwani, Qianjin Feng, and Kun Huang. Identification of topological features in renal tumor microenvironment associated with patient survival. *Bioinformatics*, 34(6):1024–1030, 2018.

[16] Xiuyuan Cheng, Qiang Qiu, Robert Calderbank, and Guillermo Sapiro. Rotdcf: Decomposition of convolutional filters for rotation-equivariant deep networks. In *International Conference on Learning Representations 2019 (ICLR'19)*, 2019.

[17] Philip Chikontwe, Meejeong Kim, Soo Jeong Nam, Heounjeong Go, and Sang Hyun Park. Multiple instance learning with center embeddings for histopathology classification. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part V 23*, pages 519–528. Springer, 2020.

[18] Kin-Hoe Chow, Rachel E Factor, and Katharine S Ullman. The nuclear envelope environment and its cancer connections. *Nature Reviews Cancer*, 12(3):196–209, 2012.

[19] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[20] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[21] Taco S Cohen and Max Welling. Steerable cnns. *arXiv preprint arXiv:1612.08498*, 2016.

[22] Lee AD Cooper, Alexis B Carter, Alton B Farris, Fusheng Wang, Jun Kong, David A Gutman, Patrick Widener, Tony C Pan, Sharath R Cholleti, Ashish Sharma, et al. Digital pathology: Data-intensive frontier in medical imaging. *Proceedings of the IEEE*, 100(4):991–1003, 2012.

[23] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[24] Corinna Cortes, Xavier Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. Adanet: Adaptive structural learning of artificial neural networks. In *International conference on machine learning*, pages 874–883. PMLR, 2017.

[25] Hengfei Cui, Lei Jiang, Chang Yuwen, Yong Xia, and Yanning Zhang. Deep u-net architecture with curriculum learning for myocardial pathology segmentation in multi-sequence cardiac magnetic resonance images. *Knowledge-Based Systems*, page 108942, 2022.

[26] Carsten Denkert, Sibylle Loibl, Aurelia Noske, Marc Roller, BM Muller, Martina Komor, Jan Budczies, Silvia Darb-Esfahani, Ralf Kronenwett, Claus Hanusch, et al. Tumor-associated lymphocytes as an independent predictor of response to neoadjuvant chemotherapy in breast cancer. *J Clin Oncol*, 28(1):105–113, 2010.

[27] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, and Xiaohua Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations*, 2021.

[28] Qi Dou, Hao Chen, Yueming Jin, Lequan Yu, Jing Qin, and Pheng-Ann Heng. 3d deeply supervised network for automatic liver segmentation from ct volumes. In *International conference on medical image computing and computer-assisted intervention*, pages 149–157. Springer, 2016.

[29] Scott Fahlman and Christian Lebiere. The cascade-correlation learning architecture. *Advances in neural information processing systems*, 2, 1989.

[30] William T Freeman, Edward H Adelson, et al. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906, 1991.

[31] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[32] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[33] Mudasir A Ganaie, Minghui Hu, AK Malik, M Tanveer, and PN Suganthan. Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115:105151, 2022.

[34] Mars L Gao, Guang Lin, and Wei Zhu. Deformation robust roto-scale-translation equivariant cnns. *Transactions on Machine Learning Research*, 2022.

[35] Yunhe Gao, Mu Zhou, Di Liu, Zhennan Yan, Shaoting Zhang, and Dimitris N Metaxas. A data-scalable transformer for medical image segmentation: architecture, model efficiency, and benchmark. *arXiv preprint arXiv:2203.00131*, 2022.

[36] J George, E Gkousis, A Feast, S Morris, S Pollard, and J Vohra. Estimating the cost of growing the nhs cancer workforce in england by 2029. *Report for Cancer Research UK*, 2020.

[37] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[38] Adam Goode, Benjamin Gilbert, Jan Harkes, Drazen Jukic, and Mahadev Satyanarayanan. Openslide: A vendor-neutral software foundation for digital pathology. *Journal of pathology informatics*, 4(1):27, 2013.

[39] Simon Graham, David Epstein, and Nasir Rajpoot. Rota-net: Rotation equivariant network for simultaneous gland and lumen segmentation in colon histology images. In *European Congress on Digital Pathology*, pages 109–116. Springer, 2019.

[40] Simon Graham, David Epstein, and Nasir Rajpoot. Dense steerable filter cnns for exploiting rotational symmetry in histology images. *IEEE Transactions on Medical Imaging*, 39(12):4124–4136, 2020.

[41] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[42] Nigel Hawkes. Cancer survival data emphasise importance of early diagnosis, 2019.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[44] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[45] Furong Huang, Jordan Ash, John Langford, and Robert Schapire. Learning deep resnet blocks sequentially using boosting theory. In *International Conference on Machine Learning*, pages 2058–2067. PMLR, 2018.

[46] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[47] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[48] Fabian Isensee, Paul F Jaeger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. nnu-net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods*, 18(2):203–211, 2021.

[49] Ziwei Ji and Matus Telgarsky. Directional convergence and alignment in deep learning. *Advances in Neural Information Processing Systems*, 33:17176–17186, 2020.

[50] Angjoo Kanazawa, Abhishek Sharma, and David Jacobs. Locally scale-invariant convolutional neural networks. *arXiv preprint arXiv:1412.5104*, 2014.

[51] Mahendra Khened, Avinash Kori, Haran Rajkumar, Balaji Srinivasan, and Ganapathy Krishnamurthi. A generalized deep learning framework for whole-slide image segmentation and analysis. *arXiv preprint arXiv:2001.00258*, 2020.

[52] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[53] Iasonas Kokkinos. Pushing the boundaries of boundary detection using deep learning. *arXiv preprint arXiv:1511.07386*, 2015.

[54] Bin Kong, Xin Wang, Zhongyu Li, Qi Song, and Shaoting Zhang. Cancer metastasis detection via spatially structured deep network. In *Information Processing in Medical Imaging: 25th International Conference, IPMI 2017, Boone, NC, USA, June 25-30, 2017, Proceedings 25*, pages 236–248. Springer, 2017.

[55] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[56] R Krithiga and P Geetha. Breast cancer detection, segmentation and classification on histopathology images analysis: a systematic review. *Archives of Computational Methods in Engineering*, 28:2607–2619, 2021.

[57] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[58] Neeraj Kumar, Ruchika Verma, Deepak Anand, Yanning Zhou, Omer Fahri Onder, Efstratios Tsougenis, Hao Chen, Pheng-Ann Heng, Jiahui Li, Zhiqiang Hu, et al. A multi-organ nucleus segmentation challenge. *IEEE transactions on medical imaging*, 39(5):1380–1391, 2019.

[59] Gustaf Kylberg. *Kylberg Texture Dataset v. 1.0*. Centre for Image Analysis, Swedish University of Agricultural Sciences and . . . , 2011.

[60] Maxime W Lafarge, Erik J Bekkers, Josien PW Pluim, Remco Duits, and Mitko Veta. Roto-translation equivariant convolutional networks: Application to histopathology image analysis. *Medical Image Analysis*, 68:101849, 2021.

[61] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pages 473–480, 2007.

[62] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[63] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436–444, 2015.

[64] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial intelligence and statistics*, pages 562–570, 2015.

[65] Joshua Levy, Christian Haudenschild, Clark Barwick, Brock Christensen, and Louis Vaickus. Topological feature extraction and visualization of whole slide images using graph neural networks. In *BIOCOMPUTING 2021: Proceedings of the Pacific Symposium*, pages 285–296. World Scientific, 2020.

[66] Jiahui Li, Shuang Yang, Xiaodi Huang, Qian Da, Xiaoqun Yang, Zhiqiang Hu, Qi Duan, Chaofu Wang, and Hongsheng Li. Signet ring cell detection with a semi-supervised learning framework. In *International conference on information processing in medical imaging*, pages 842–854. Springer, 2019.

[67] Yi Li and Wei Ping. Cancer metastasis detection with neural conditional random field. In *Medical Imaging with Deep Learning*, 2018.

[68] Tony Lindeberg. Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of applied statistics*, 21(1-2):225–270, 1994.

[69] Tony Lindeberg. Feature detection with automatic scale selection. *International journal of computer vision*, 30:79–116, 1998.

[70] Tony Lindeberg. Scale-covariant and scale-invariant gaussian derivative networks. *Journal of Mathematical Imaging and Vision*, 64(3):223–242, 2022.

[71] Jasper Linmans, Jim Winkens, Bastiaan S Veeling, Taco S Cohen, and Max Welling. Sample efficient semantic segmentation using rotation equivariant convolutional networks. *arXiv preprint arXiv:1807.00583*, 2018.

[72] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[73] Haozhe Luo, Yu Changdong, and Raghavendra Selvan. Hybrid ladder transformers with efficient parallel-cross attention for medical image segmentation. In *Medical Imaging with Deep Learning*, 2022.

[74] Mingjun Ma, Haiying Xia, Yumei Tan, Haisheng Li, and Shuxiang Song. Ht-net: hierarchical context-attention transformer network for medical ct image segmentation. *Applied Intelligence*, pages 1–14, 2022.

[75] Marc Macenko, Marc Niethammer, James S Marron, David Borland, John T Woosley, Xiaojun Guan, Charles Schmitt, and Nancy E Thomas. A method for normalizing histology slides for quantitative analysis. In *2009 IEEE international symposium on biomedical imaging: from nano to macro*, pages 1107–1110. IEEE, 2009.

[76] Diego Marcos, Benjamin Kellenberger, Sylvain Lobry, and Devis Tuia. Scale equivariance in cnns with vector fields. *arXiv preprint arXiv:1807.11783*, 2018.

[77] Enrique S Marquez, Jonathon S Hare, and Mahesan Niranjan. Deep cascade learning. *IEEE transactions on neural networks and learning systems*, 29(11):5475–5485, 2018.

[78] J Martin. Meeting pathology demand: Histopathology workforce census. *London: Royal College of Pathologists*, 2018.

[79] Caner Mercan, Selim Aksoy, Ezgi Mercan, Linda G Shapiro, Donald L Weaver, and Joann G Elmore. From patch-level to roi-level deep feature representations for breast histopathology classification. In *Medical Imaging 2019: Digital Pathology*, volume 10956, pages 86–93. SPIE, 2019.

[80] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3523–3542, 2021.

[81] Nikita Moshkov, Botond Mathe, Attila Kertesz-Farkas, Reka Hollandi, and Peter Horvath. Test-time augmentation for deep learning-based cell segmentation on microscopy images. *Scientific reports*, 10(1):5068, 2020.

[82] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

[83] World Health Organization. International agency for research on cancer, globocan 2020. , 2020. Accessed: 2023-06-01.

[84] Sebastian Otálora, Manfredo Atzori, Vincent Andrearczyk, and Henning Müller. Image magnification regression using densenet for exploiting histopathology open access content. In *Computational Pathology and Ophthalmic Medical Image Analysis: First International Workshop, COMPAY 2018, and 5th International Workshop, OMIA 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16-20, 2018, Proceedings 5*, pages 148–155. Springer, 2018.

[85] PAIP Organizers. Technical reviews on paip2019 challenge, 2019. URL . [Online; accessed 26-May-2020].

[86] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[87] Silvia L Pintea, Nergis Tömen, Stanley F Goes, Marco Loog, and Jan C van Gemert. Resolution learning in deep convolutional networks using scale-space theory. *IEEE Transactions on Image Processing*, 30:8342–8353, 2021.

[88] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[89] Abhijit Guha Roy, Nassir Navab, and Christian Wachinger. Concurrent spatial and channel 'squeeze & excitation'in fully convolutional networks. In *International conference on medical image computing and computer-assisted intervention*, pages 421–429. Springer, 2018.

[90] Raphael Rubin, David S Strayer, Emanuel Rubin, et al. *Rubin's pathology: clinicopathologic foundations of medicine*. Lippincott Williams & Wilkins, 2008.

[91] Rüdiger Schmitz, Frederic Madesta, Maximilian Nielsen, Jenny Krause, Stefan Steurer, René Werner, and Thomas Rösch. Multi-scale fully convolutional neural networks for histopathology image segmentation: from nuclear aberrations to the global tissue architecture. *Medical image analysis*, 70:101996, 2021.

[92] Wei Shao, Tongxin Wang, Zhi Huang, Zhi Han, Jie Zhang, and Kun Huang. Weakly supervised deep ordinal cox model for survival prediction from whole-slide pathological images. *IEEE Transactions on Medical Imaging*, 40(12):3739–3747, 2021.

[93] Jiangbo Shi, Lufei Tang, Yang Li, Xianli Zhang, Zeyu Gao, Yefeng Zheng, Chunbao Wang, Tieliang Gong, and Chen Li. A structure-aware hierarchical graph-based multiple instance learning framework for pt staging in histopathological image. *IEEE Transactions on Medical Imaging*, 2023.

[94] Korsuk Sirinukunwattana, Josien PW Pluim, Hao Chen, Xiaojuan Qi, Pheng-Ann Heng, Yun Bo Guo, Li Yang Wang, Bogdan J Matuszewski, Elia Bruni, Urko Sanchez, et al. Gland segmentation in colon histology images: The glas challenge contest. *Medical image analysis*, 35:489–502, 2017.

[95] Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, volume 11006, page 1100612. International Society for Optics and Photonics, 2019.

[96] Ivan Sosnovik, Michał Szmaja, and Arnold Smeulders. Scale-equivariant steerable networks. In *International Conference on Learning Representations*, 2020.

[97] Ivan Sosnovik, Artem Moskalev, and Arnold Smeulders. How to transform kernels for scale-convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 1092–1097, October 2021.

[98] Ivan Sosnovik, Artem Moskalev, and Arnold Smeulders. Disco: accurate discrete scale convolutions. *arXiv preprint arXiv:2106.02733*, 2021.

[99] Manuel Stritt, Anna K Stalder, and Enrico Vezzali. Orbit image analysis: an open-source whole slide image analysis tool. *PLoS computational biology*, 16(2): e1007313, 2020.

[100] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[101] Aboozar Taherkhani, Georgina Cosma, and T Martin McGinnity. Adaboost-cnn: An adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning. *Neurocomputing*, 404: 351–366, 2020.

[102] Abhishek Vahadane, Tingying Peng, Amit Sethi, Shadi Albarqouni, Lichao Wang, Maximilian Baust, Katja Steiger, Anna Melissa Schlitter, Irene Esposito, and Nassir Navab. Structure-preserving color normalization and sparse stain separation

for histological images. *IEEE transactions on medical imaging*, 35(8):1962–1971, 2016.

[103] Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant cnns for digital pathology. In *International Conference on Medical image computing and computer-assisted intervention*, pages 210–218. Springer, 2018.

[104] Mitko Veta, Paul J Van Diest, Mehdi Jiwa, Shaimaa Al-Janabi, and Josien PW Pluim. Mitosis counting in breast cancer: Object-level interobserver agreement and comparison to an automatic method. *PloS one*, 11(8):e0161286, 2016.

[105] Surbhi Vijh, Mukesh Saraswat, and Sumit Kumar. A new complete color normalization method for h&e stained histopatholgical images. *Applied Intelligence*, pages 1–14, 2021.

[106] Dayong Wang, Aditya Khosla, Rishab Gargeya, Humayun Irshad, and Andrew H Beck. Deep learning for identifying metastatic breast cancer. *arXiv preprint arXiv:1606.05718*, 2016.

[107] Jianlian Wang, Jun Ruan, Simin He, Chenchen Wu, Guanglu Ye, Jingfan Zhou, Junqiu Yue, and Yanggeling Zhang. Detection of her2 scores and magnification from whole slide images in multi-task convolutional network. In *2018 11th International Symposium on Computational Intelligence and Design (ISCID)*, volume 1, pages 7–10. IEEE, 2018.

[108] Junwen Wang, Katayoun Farrahi, and Mahesan Niranjan. Curriculum based cascade learning for image classification. 2020.

[109] Xi Wang, Hao Chen, Caixia Gan, Huangjing Lin, Qi Dou, Efstratios Tsougenis, Qitao Huang, Muyan Cai, and Pheng-Ann Heng. Weakly supervised deep learning for whole slide lung cancer image analysis. *IEEE transactions on cybernetics*, 50(9):3950–3962, 2019.

[110] Yulin Wang, Zanlin Ni, Shiji Song, Le Yang, and Gao Huang. Revisiting locally supervised learning: an alternative to end-to-end training. *arXiv preprint arXiv:2101.10832*, 2021.

[111] Maurice Weiler and Gabriele Cesa. General E(2)-Equivariant Steerable CNNs. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[112] Maurice Weiler and Gabriele Cesa. General e (2)-equivariant steerable cnns. *Advances in neural information processing systems*, 32, 2019.

[113] Maurice Weiler, Fred A Hamprecht, and Martin Storath. Learning steerable filters for rotation equivariant cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 849–858, 2018.

[114] Daniel Worrall and Max Welling. Deep scale-spaces: Equivariance over scale. *Advances in Neural Information Processing Systems*, 32, 2019.

[115] Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5028–5037, 2017.

[116] Yawen Wu, Michael Cheng, Shuo Huang, Zongxiang Pei, Yingli Zuo, Jianxin Liu, Kai Yang, Qi Zhu, Jie Zhang, Honghai Hong, et al. Recent advances of deep learning for computational histopathology: Principles and applications. *Cancers*, 14(5):1199, 2022.

[117] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[118] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015.

[119] Yan Xu, Zhipeng Jia, Liang-Bo Wang, Yuqing Ai, Fang Zhang, Maode Lai, Eric I Chang, et al. Large scale tissue histopathology image classification, segmentation, and visualization via deep convolutional activation features. *BMC bioinformatics*, 18(1):1–17, 2017.

[120] Yichong Xu, Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, and Zheng Zhang. Scale-invariant convolutional neural networks. *arXiv preprint arXiv:1411.6369*, 2014.

[121] Yilong Yang, Srinandan Dasmahapatra, and Sasan Mahmoodi. Scale-equivariant unet for histopathology image segmentation. In *Proceedings of the First International Workshop on Geometric Deep Learning in Medical Image Analysis*, volume 194 of *Proceedings of Machine Learning Research*, pages 130–148. PMLR, 18 Nov 2022. URL .

[122] Jiawen Yao, Xinliang Zhu, Jitendra Jonnagaddala, Nicholas Hawkins, and Junzhou Huang. Whole slide images based cancer survival prediction using attention guided deep multiple instance learning networks. *Medical Image Analysis*, 65: 101789, 2020.

[123] Farhad Ghazvinian Zanjani, Svitlana Zinger, et al. Cancer detection in histopathology whole-slide images using conditional random fields on deep embedded spaces. In *Medical imaging 2018: Digital pathology*, volume 10581, pages 143–149. SPIE, 2018.

[124] Manit Zaveri, Shivam Kalra, Morteza Babaie, Sultaan Shah, Savvas Damskinos, Hany Kashani, and Hamid R Tizhoosh. Recognizing magnification levels in microscopic snapshots. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, pages 1416–1419. IEEE, 2020.

[125] Hongtai Zhang, Zaiyi Liu, Mingli Song, and Cheng Lu. Hagnifinder: Recovering magnification information of digital histological images using deep learning. *Journal of Pathology Informatics*, 14:100302, 2023.

[126] Quan Zhou, Xiaofu Wu, Suofei Zhang, Bin Kang, Zongyuan Ge, and Longin Jan Latecki. Contextual ensemble network for semantic segmentation. *Pattern Recognition*, 122:108290, 2022.

[127] Zhi-Hua Zhou and Zhi-Hua Zhou. *Ensemble learning*. Springer, 2021.

[128] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: Redesigning skip connections to exploit multiscale features in image segmentation. *IEEE transactions on medical imaging*, 39(6):1856–1867, 2019.

[129] Qikui Zhu, Bo Du, Baris Turkbey, Peter L Choyke, and Pingkun Yan. Deeply-supervised cnn for prostate segmentation. In *2017 International Joint Conference on Neural Networks (Ijcnn)*, pages 178–184. IEEE, 2017.

[130] Wei Zhu, Qiang Qiu, Robert Calderbank, Guillermo Sapiro, and Xiuyuan Cheng. Scaling-translation-equivariant networks with decomposed convolutional filters. *Journal of Machine Learning Research*, 23(68):1–45, 2022.

[131] Xinliang Zhu, Jiawen Yao, and Junzhou Huang. Deep convolutional neural network for survival analysis with pathological images. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 544–547. IEEE, 2016.