

# A SYSTEM FOR ANALYSING AXON ACTIVITY USING MULTI-ANGLE AND SCALE CONVOLUTIONAL NETWORKS

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER  
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY  
IN THE FACULTY OF BIOLOGY, MEDICINE AND HEALTH

2022

Zewen Liu

Division of Informatics, Imaging and Data Sciences  
School of Health Sciences

# Contents

<b>Abstract</b>	<b>10</b>
<b>Declaration</b>	<b>11</b>
<b>Copyright</b>	<b>12</b>
<b>Publications arising from this work</b>	<b>13</b>
<b>Acknowledgements</b>	<b>14</b>
<b>1 Introduction</b>	<b>15</b>
1.1 Challenges and Solutions . . . . .	16
1.2 Thesis Structure . . . . .	17
<b>2 Literature Review</b>	<b>19</b>
2.1 What Is Needed for Segmentation . . . . .	20
2.1.1 A Brief Introduction to Convolutional Neural Networks . . . . .	20
2.1.2 Segmentation Methods . . . . .	22
2.2 Matched Filters . . . . .	33
2.3 Attention in Image Processing . . . . .	34
2.3.1 Vision Transformers . . . . .	36
2.4 Rotational Symmetric Pattern . . . . .	38
2.4.1 Steerable CNN . . . . .	38
2.4.2 G-CNNs . . . . .	39
2.4.3 Harmonic Basis Methods . . . . .	39
2.5 Modelling Structured Data . . . . .	40
2.5.1 Recurrent Models for Sequential Data . . . . .	41
2.6 Graph Neural Networks . . . . .	42
2.6.1 Recurrent Graph Neural Networks . . . . .	43



2.6.2	Graph Convolutional Neural Networks . . . . .	44
2.6.3	Axon Tracking Techniques . . . . .	47
<b>3</b>	<b>Multi Angle and Scale Convolution</b>	<b>50</b>
3.1	Introduction . . . . .	50
3.2	Algorithm . . . . .	53
3.2.1	Multi-Angle Convolution: Introduction . . . . .	53
3.2.2	MAC: Hybrid Convolution . . . . .	54
3.2.3	MAC: Kernel Reuse . . . . .	59
3.2.4	MAC: Response Shaping . . . . .	62
3.2.5	MAC: Update $\mathbf{M}$ with Momentum . . . . .	67
3.2.6	MASC: Pyramid Representation . . . . .	67
3.2.7	MASC: Re-Scale Feature Maps . . . . .	68
3.2.8	Building Block Construction . . . . .	72
3.3	Model Structure . . . . .	73
3.4	Experiments . . . . .	74
3.4.1	Datasets . . . . .	74
3.4.2	Configuration . . . . .	75
3.4.3	Rotation Evaluation . . . . .	75
3.4.4	Other Evaluation Metrics . . . . .	76
3.4.5	General Performance . . . . .	77
3.4.6	Hyper-Parameter Studies . . . . .	78
3.5	Discussion . . . . .	91
3.5.1	Index Maps . . . . .	91
3.5.2	Memory Optimisation Choice . . . . .	92
3.6	Conclusion . . . . .	92
<b>4</b>	<b>General MASC</b>	<b>94</b>
4.1	Introduction . . . . .	94
4.2	Motivations . . . . .	96
4.2.1	Feasibility Discussion . . . . .	96
4.2.2	Challenges . . . . .	96
4.2.3	Rotation Symmetry Supervision . . . . .	98
4.2.4	Direction and Scale . . . . .	100
4.3	Algorithm . . . . .	102
4.3.1	Kernel Scheme . . . . .	102

4.3.2	Calibrated Response Shaping . . . . .	103
4.3.3	Index Map Featurisation . . . . .	103
4.3.4	Rotational Penalty . . . . .	106
4.3.5	Why use squared residual but not correlation? . . . . .	108
4.3.6	Model Structure . . . . .	109
4.4	Experiment . . . . .	110
4.4.1	Environment . . . . .	110
4.4.2	General Performance . . . . .	111
4.4.3	Ablation Study . . . . .	114
4.4.4	PoRE Rotational Cost . . . . .	115
4.4.5	Compare PoRE with PoC . . . . .	118
4.4.6	Update $M$ in Different Ways . . . . .	118
4.4.7	Kernel Construction . . . . .	119
4.4.8	Kernel Initialisation . . . . .	120
4.5	Conclusion . . . . .	121
<b>5</b>	<b>Tracking Axon: Model-1</b>	<b>122</b>
5.1	General Plan . . . . .	123
5.2	Feature Extraction . . . . .	125
5.2.1	First Round Cleaning . . . . .	126
5.2.2	Building the Primary Axon Profile . . . . .	127
5.2.3	The Cell Profile . . . . .	130
5.2.4	Correctly Determining Unique Splitting Nodes . . . . .	130
5.3	Heat Map Based Tracking Module . . . . .	133
5.3.1	History Encoding . . . . .	134
5.3.2	Task 2 Solution: Complete Axon Trace with Two-Stage Checks	135
5.3.3	Task 3 Solution: Iterative Connect&Cut . . . . .	142
5.3.4	Repair Mechanism . . . . .	149
5.3.5	Determine Initial Segment, Cell and Root . . . . .	152
5.3.6	Stop Tracking Conditions . . . . .	154
5.4	Conclusion . . . . .	156
<b>6</b>	<b>Experiment Analysis and Model-2</b>	<b>157</b>
6.1	Drug Experiment Analysis . . . . .	157
6.1.1	Experiment Introduction . . . . .	158
6.1.2	Strip Recognition . . . . .	158

6.1.3	Video Stablisation . . . . .	160
6.1.4	Interaction Definition . . . . .	160
6.1.5	Measurement . . . . .	160
6.1.6	Results . . . . .	161
6.1.7	Discussion . . . . .	166
6.2	Model 2: AxonTree Model . . . . .	167
6.2.1	Background . . . . .	167
6.2.2	Dataset Construction . . . . .	167
6.2.3	Feature Matrix . . . . .	168
6.2.4	Edge Direction Matrix . . . . .	169
6.2.5	Methodolgy . . . . .	169
6.2.6	Performance Comparison . . . . .	171
6.2.7	Conclusion . . . . .	171
<b>7</b>	<b>Conclusion</b>	<b>172</b>
7.1	Contributions . . . . .	172
7.2	Future Works . . . . .	173
	<b>Bibliography</b>	<b>175</b>

**Word Count: 41402**

# List of Tables

2.1	CNN architectures . . . . .	21
3.1	Segmentation Results on CHASE_DB1 . . . . .	79
3.2	Segmentation Results on DRIVE . . . . .	79
3.3	Segmentation Results on Axon Dataset . . . . .	79
3.4	Module Ablation Results . . . . .	80
3.5	Parallel units numbers . . . . .	82
3.6	MAC direction numbers . . . . .	84
3.7	MAC kernel sizes . . . . .	85
3.8	Key matrix $\mathbf{M}$ with momentums . . . . .	85
3.9	Kernel Construction and Initialisation . . . . .	88
3.10	Response Shaping, Max Response, and Squared Response Difference	89
3.11	MASC vs UNet . . . . .	90
4.1	G-MASC general results . . . . .	113
4.2	Ablation study . . . . .	114
4.3	Cost function weights . . . . .	117
4.4	Kernel size and number of references . . . . .	118
4.5	PoC vs PoRE . . . . .	118
4.6	Momentum study . . . . .	119
4.7	Kernel initialisation . . . . .	120
4.8	Kernel initialisation . . . . .	120
6.1	Tracking models comparison . . . . .	170

# List of Figures

1.1	Axons, blobs, and neural cells under microscope. . . . .	15
1.2	Graphical overview of this thesis . . . . .	17
2.1	Fully connected network . . . . .	23
2.2	Conditional random field FCN . . . . .	23
2.3	SegNet architecture . . . . .	25
2.4	U-Net architecture . . . . .	27
2.5	Dilated Convolution . . . . .	28
2.6	Recursive semantic-guidance network . . . . .	30
2.7	Relation Network . . . . .	32
2.8	Attention Mechanism . . . . .	34
2.9	Non-local network . . . . .	35
2.10	Vision transformer . . . . .	37
2.11	Steerable CNN . . . . .	39
2.12	Harmonic basis . . . . .	40
2.13	RNN, LSTM and GRU . . . . .	42
2.14	Spectral GCN . . . . .	45
2.15	Reinforcement axon tracing . . . . .	48
3.1	Isotropic patterns . . . . .	52
3.2	Concepts introduction . . . . .	53
3.3	Gabor function demonstration . . . . .	56
3.4	The patterns of trained hybrid kernels . . . . .	57
3.5	Hybrid kernel construction . . . . .	59
3.6	The impact of using different numbers of angles in MASC networks . . . . .	61
3.7	Directional kernels and $\mathbf{M}$ matrix . . . . .	63
3.8	MASC outputs visualisation . . . . .	65
3.9	Parallel and serial processing in MASC networks . . . . .	68

3.10	Comparison between the two re-scale schemes . . . . .	69
3.11	Re-scale case study . . . . .	71
3.12	MASC model structures . . . . .	73
3.13	Segmentation results on CHASE-DB1 . . . . .	78
3.14	Ablation experiment results . . . . .	81
3.15	Performance using different numbers of parallel units in a block . . . . .	83
3.16	Key matrix $\mathbf{M}$ with momentums . . . . .	86
3.17	MASC vs UNet . . . . .	90
4.1	G-MASC motivations . . . . .	97
4.2	Index map featuralisation . . . . .	104
4.3	Response shaping and PoRE . . . . .	106
4.4	G-MASC unit, block, and model structure . . . . .	109
4.5	Training samples MoNuSeg . . . . .	110
4.6	The segmentation results on CHASE-DB1 and DRIVE . . . . .	112
4.7	The segmentation results on MoNuSeg . . . . .	113
4.8	Sensitivities of Lambda Choices . . . . .	116
5.1	Axon tracking challenges . . . . .	123
5.2	Implementation plan . . . . .	124
5.3	Blob fixation . . . . .	126
5.4	First data clean . . . . .	127
5.5	Primary profile . . . . .	128
5.6	Node determination issues . . . . .	131
5.7	Tracking model-1 results . . . . .	132
5.8	Activity heatmap construction . . . . .	136
5.9	Problems solved during the Model-1 process . . . . .	137
5.10	Rough searching . . . . .	138
5.11	Edge adjustment explanation . . . . .	141
5.12	Repair and prune procedures . . . . .	144
5.13	Manual repair and prune procedures . . . . .	147
5.14	Repair operation demonstration . . . . .	150
5.15	Creating initial main tree procedures . . . . .	153
5.16	Model-1 tracking examples . . . . .	156
6.1	Conception explanation . . . . .	158
6.2	Strip pattern extraction . . . . .	159

6.3	Interaction statuses explanation . . . . .	161
6.4	Axon length against time when interacting different types of strip patterns	162
6.5	Axon case study with interaction statuses indicated . . . . .	163
6.6	Axon growth pattern studied by the interaction statuses . . . . .	164
6.7	Axon growth pattern studies in first 30 and 50 frames with t-SNE and principle components analysis . . . . .	165
6.8	Axon edge direction expectation measurement . . . . .	169
6.9	Tracking model-2 architecture . . . . .	170
6.10	Model-2 is also able to detach traversing axons and reconnect broken traces. The left plot is a binary segmentation map, the right plot shows the corresponding tracking output in which the left bottom axon is se- lected as the target object. . . . .	170

# Abstract

## A SYSTEM FOR ANALYSING AXON ACTIVITY USING MULTI-ANGLE AND SCALE CONVOLUTIONAL NETWORKS

Zewen Liu

A thesis submitted to The University of Manchester  
for the degree of Doctor of Philosophy, 2022

To study how axon growth is affected by the local environment biologists perform extensive experiments, watching the axons develop on different substrates. As axons grow from the neuron cell body they form tree-like structures, with branches forming and withering as they explore their surroundings.

In this project, we designed a system which can track individual axons as they grow and branch over time, enabling quantitative evaluation of different aspects of the axon behaviour.

The system includes a novel segmentation network that can be aware of not only intensity but also properties such as feature direction and scale. Such properties can be important when analysing images containing curvilinear structures such as vessels or fibres. We propose the General Multi-Angle Scale Convolution (G-MASC) network, whose kernels are arbitrarily rotatable and also fully differentiable. The model manages its directional detectors in sets, and supervises a set's rotational symmetry with a novel rotation penalty called PoRE. The algorithm works on pyramid representations to enable scale search. Direction and scale can be extracted from the output maps, encoded and analysed separately. Tests were conducted on three public datasets and the axon samples. Good performance is observed while the model requires 1% or fewer parameters compared to other approaches such as U-Net.



# Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

# Copyright

- i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the “Copyright”) and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.
- ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made **only** in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.
- iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the “Intellectual Property”) and any reproductions of copyright works in the thesis, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.
- iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see <http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420>), in any relevant Thesis restriction declarations deposited in the University Library, The University Library’s regulations (see <http://www.library.manchester.ac.uk/about/regulations/>) and in The University’s policy on presentation of Theses

# Publications arising from this work

## 1. **An end to end system for measuring axon growth**

Liu, Zewen, Timothy Cootes, and Christoph Ballestrem. International Workshop on Machine Learning in Medical Imaging 2020, pp. 455-464 (MICCAI-MLMI 2020)

## 2. **MASC-Units: Training Oriented Filters for Segmenting Curvilinear Structures**

Liu, Zewen, and Timothy Cootes. International Conference on Medical Image Computing and Computer-Assisted Intervention 2021, pp. 590-599 (MICCAI 2021)

## 3. **A Sense of Direction in Biomedical Neural Networks**

Liu, Zewen, and Timothy Cootes. International Conference on Medical Image Computing and Computer-Assisted Intervention 2022 (MICCAI 2022)

# Acknowledgements

I would like to thank my supervisor Prof. Timothy F. Cootes for the enormous amount of support across my project. He imparted me the knowledge of math and algorithms, and more importantly, the disciplines about becoming a decent researcher by his everyday practice. I would like to thank my co-supervisors, Prof. Christoph Ballestrem for all the help and guidance - especially in the biological aspect of the project. The microscopy data of axons used in the tests of this project was provided by Christoph Ballestrem Lab collected in the University of Manchester

Most of this project was during the COVID pandemic. I would also like to thank my mother and Yue Yu for helping me getting through the physically, economically and mentally tough period.

# Chapter 1

## Introduction

When studying how neurons develop, biologists perform many experiments in which they collect videos of cells as they grow. To analyse these they must track the tips of the growing axons as they grow across a medium in their search for other cells to connect to. As shown in Fig. 1.1, the axon tip can be tiny. Studying the growth activities of axons is crucial for answering the what and how questions in neural cell response. For example, what response should be expected when given a specific type of biomedical cue. This is important in surgical recovery.

Currently, this tracking involves the experimenter clicking on the end of the axon in each of many frames – a very time-consuming process. The original goal of this thesis was to automate this process, to speed it up and to enable much larger scaled and more detailed analysis of the microscopy videos that have been collected.

Axons are thin curvilinear structures. In order to segment them, we developed a

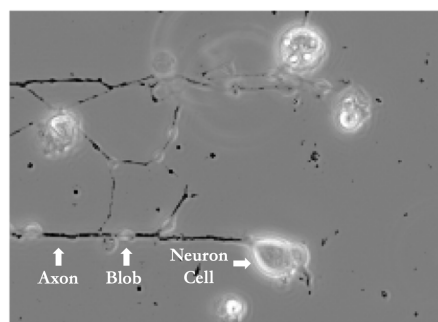


Figure 1.1: Axons, blobs, and neural cells under microscope. Axons are the black branch-like structures growing out of neuron cells. There are some small blobs sticking to them, whose function remains mysterious. These objects are collected and studied in this system.

novel network architecture based on constructing a set of filters designed to be approximately invariant to rotation (Multi-Angle and Scale Convolution – MASC [80]). The resulting network has proved to be very efficient, achieving results comparable to state-of-the-art CNNs such as U-Net but using fewer than 1% of the parameters. We went on to generalise the approach (G-MASC) and show it could work on a range of segmentation problems.

## 1.1 Challenges and Solutions

Processing video sequences of growing axons requires three main steps:

1. Identifying, in each frame, the pixels belonging to axons (segmentation)
2. Linking nearby axon pixels to find the shape of a whole axon, and tracking it over time through video
3. Analysing the way an axon grows and interacts with the substrate on which it is growing.

### Segmenting the axons

Convolutional Neural Networks (CNNs) currently provide the most effective methods for segmenting structures in images. CNNs contain banks of filters in each layer, which are trained to pick out features useful for the segmentation task. However, the random initialisation in a typical model produces parameter redundancy. The nature of axon images is special. There are many rotational and scale variants of the same patterns in these images. A compact model was designed, which takes the advantage of the textures of the known structures. Such a MASC model uses the same set of detectors in multiple orientations and scales. Unlike other rotational-invariant algorithms, the directional detectors in our model are fully trainable and can have arbitrary orientations.

The MASC model was designed for axon patterns. However, there are many very different patterns in biomedical images. In order to make the modelling for these objects as sufficient as for axons, the G-MASC was proposed. It overcame the issues in the original algorithm and transferred the middle outputs into useful new features.

**Instance Locating** Axons can be entangled as shown in Fig. 1.1. To separate them, areas were refined iteratively. Pieces from consecutive frames were linked to generate a sequence of axon activity in video time.

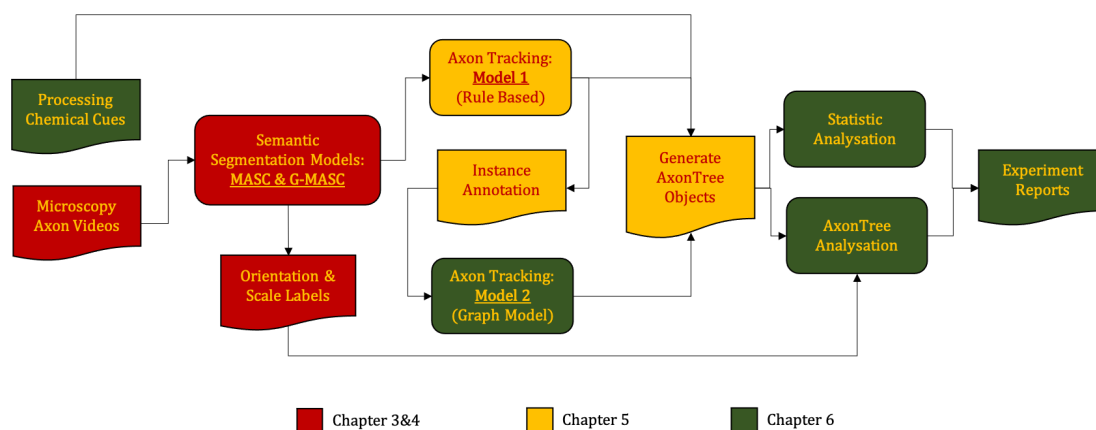


Figure 1.2: Graphical overview of work in this thesis. Topics are coloured according to the corresponding chapter.

**Analysing Axons:** The current approach used by biologists, that of manually tracking the axon tip, yields only limited information about the whole of the structure. Our system can automatically locate and measure the entire axon tree, enabling many different features to be calculated (such as axon length, numbers of branches and so on). This also increased the modelling complexity. We introduced an abstract AxonTree structure to represent axons, so the morphological factors can be integrated. Graph methods were also applied to these trees to support our analysis.

## 1.2 Thesis Structure

The thesis has three major topics, which are segmentation, tracking (axon locating), and data analysis. In Fig.1.2, the general system structure is presented, in which the modules were coloured according to the related chapters.

### Chapter 2: Literature Review

Our focus is on building a segmentation model and strengthening the model's performance in several ways including receptive field and rotational-invariant detection. These algorithms inspired us when developing the model of MASC. A brief introduction about recurrent networks and graph networks is included, which is related to the data analysis module.

### Chapter 3: Axon Segmentation - Multi Angle and Scale Convolution

We designed a semantic segmentation algorithm for finding axon area in images called MASC [80].

We studied the axon images and found most objects can be described as variants of one pattern in different orientations and widths. Building on this observation, MASC uses directional kernel sets to extract rotational-invariant information from feature maps. The directional kernels are updated so as to retain rotational symmetry. The resulting network was found to be more efficient and accurate than the U-Net benchmarks on axon and retina vessel datasets.

#### **Chapter 4: A Sense of Direction - G-MASC**

We generalised the idea of multi-angle and scale convolution by G-MASC. It addressed the underdetermined issue in MASC and helped detection by encoding the by-products (label maps) as new feature sources. Tests were made on different datasets and some interesting and encouraging results were presented.

#### **Chapter 5: Tracking Axon - Model-1**

In the tracking step, we took a two-step scheme to overcome the challenge of instance tracking, called Model-1 and Model-2. The axon objects detected by a semantic segmentation model are extracted and wrapped as AxonTrees which is a specially crafted data structure. A history-based tracking model(Model-1) was used to conduct the preliminary tracking. It is a rule based model. We designed a three-round refinement for it to disentangle the crossed axon instances in images. In practice, the output of Model-1 with correction was found to be accurate enough for data analysis.

#### **Chapter 6: Axon Analysis**

After the quality supervision, the tracking data produced by Model-1 were packed as an instance dataset for training the tracking Model-2 which is faster and more robust. This is also an example of using graph convolution for understanding axons at the node and edge level. We discussed another example of axon analysis. Biologists perform experiments to understand how axons behave in response to different chemicals on the growth substrate. We present the results of applying our automatic analysis to the resulting videos, and compare their performance to the studies from human annotators.

#### **Chapter 7: Conclusion**

We summarise our contributions and propose future directions for the research.



# Chapter 2

## Literature Review

Here we review the background literature related to the work in this thesis. It is arranged as the logic illustrated in 1.2 Chapter 1. Six topics are covered, which are segmentation algorithms, matched filters, attention models, steerable kernels, sequential modelling, and graph neural networks. Each corresponds to a system module.

Starting from the deep learning segmentation, the first section will cover the major convolutional architectures, as well as the convolutional applications in segmentation tasks. The models are introduced by category. The algorithms such as dilated searching and pyramid pooling which can effectively increase a model's receptive field will also be reviewed. The section will include descriptions about boosting local detection and instance-level segmentation. Our system is designed for subtle pattern detection, e.g. blurred axon branches. These algorithms lead to sensitive detection with noise control and inspired the idea of MASC model.

Then, we will talk about the idea of matched filters, from which we designed the fundamental logic of rotational detection.

Initially designed for language processing, the attention mechanism, and transformer modules were found helpful in solving long-range context reliance problems. Some examples have been proposed for addressing medical image tasks [113] [131]. Our algorithm borrowed the idea, where its key-query mechanism enables a collective orientational message to be generated in MASC. However, the method has drawbacks, for example, heavily relying on pre-training and needing much more computation resource than normal Convolutional Neural Networks (CNNs).

We will describe the application of steerable kernels in the culture of deep learning. Three typical kinds of them will be discussed, including steerable CNN, group CNN, and harmonic basis methods.

A good detection ensures axon targets to be accurately located in microscopy videos. Returning to the experiment’s biological objective, we need to find a proper formation to store, quantify, and analyse axons’ position, morphology, and activity record. In our system, axon instances are abstracted as trees, which capture their biological information and preserve morphological information. Since the statistical problem is converted into a graph problem, many advanced graph analysing theories can be used in this project. We will talk about how to deal with structured data effectively, either continuous or discrete. The methods for decoding valuable information for classification and prediction are described from the aspects of Recurrent Neural Networks (RNNs) and Graph Neural Networks (GNNs).

## 2.1 What Is Needed for Segmentation

### 2.1.1 A Brief Introduction to Convolutional Neural Networks

The most popular segmentation solutions today (2022) are based on convolutional neural networks (CNNs). Here we summarise the development of CNNs.

Deep learning in today’s sense usually refers to a kind of machine learning algorithm with multi-layer perceptrons structure (MLP), and using stochastic gradient descent [67] or variants as optimisation method to update flexible parameters. It was justified by analogy to human neural systems, and is also called (artificial) neural network in many scenarios. A neural network can be very deep. Depending on the complexity of a task, networks may have hundreds of layers. Recent studies have found that deeper structure does not always produce better performance [84]. From a shallow feature to a deep feature, the extracted information becomes more and more abstract, which can be over-exploited and cause overfitting, gradient vanishing, and other undesired conditions. The calculation of each perceptron is a linear transformation of the input feature vector which can be expressed as,

$$\text{MLP}(\mathbf{x}, \mathbf{w}) = \Phi\left(b + \sum_{i=1}^n \mathbf{x}_i \mathbf{w}_i\right) \quad (2.1)$$

It enables an input feature point  $\mathbf{x}$  in  $n$ -dimensional spaces to be scaled by weights  $\mathbf{w}$  and translated (by  $b$ ). Both  $\mathbf{w}$  and  $b$  are updatable parameters, which evolve along the direction of gradients during training. The notation  $\Phi$  here is a non-linear activation function. Without it, a stack of nodes would be equivalent to a single linear transformation. It helps the model to have versatile transformation thus encoding more complex

Table 2.1: CNN Architectures (Tested on ImageNet, \* on MNIST)

Architecture	Year	Para.	Top5	Characters
LeNet [68]	1998	0.060 M	0.8*	Representative early work.
AlexNet [64]	2012	60 M	11.7	1. Use GPUs 2. Deeper network 3. Use Relu, dropout, and pooling.
VGG [106]	2014	138 M	7.3	Use 3x3 kernels.
GoogLeNet [110]	2015	4 M	6.7	Use dimension-reduce module.
ResNet [48]	2016	25.6 M	3.6	Residual connection.
Residual Attention Neural Network [117]	2017	8.6 M	4.8	Use attention mechanism.
ResNeXt [130]	2017	68.1 M	4.4	Grouped convolution.
NASNet [150]	2017	88.9 M	3.8	Improve channel interdependency.
MobileNet [52]	2017	3.2M	10.5	1. GeLU 2. Invert bottleneck.
SENet [54]	2018	27.5M	2.3	Improve channel interdependency.
EfficientNet [111]	2019	66M	3	Model scaling and searching.

knowledge, stretching the distance between positive and negative feature points.

Many activation functions were studied, and sigmoid was the most popular. However, a general problem of the functions such as sigmoid and tanh is their saturation issue. Strong signals on both sides are compressed. Recently new activation functions are proposed, for example, ReLU [12], GeLU (2016) [49], ReLU6 (2017) [52], SIREN (2020) [107], TanhExp (2021) [78]. They have different characteristics in calculation efficiency and performance. The choice of activate function should depend on the nature of the feature distribution and designed so as to prevent gradient vanishing/-explosion during training as well as overfitting.

Convolutional Neural Networks are a type of network that shares kernel weights among positions. Instead of employing fully connected layers to cover every input pixel, they employ a sliding convolution window to analyse the local pattern at each position. Compared to a fully connected neural network, they require fewer parameters. A collection of  $3 \times 3$  kernels uses far fewer parameters than the matrix required for a fully connected transformation. CNNs have been shown to be very flexible. They are well suited to analysis of images.

A list of popular CNN architectures is in Table.2.1 which also includes the model scale and the top-5 error rate on the ImageNet classification task. Between network and convolutional layer, the convention of layer block was used since GoogLeNet [110], in which multiple convolutional layers are wrapped to use together. These blocks are

treated like basic building units, and a regular convolutional network today is usually constructed by combining building blocks. Features such as residual connections and narrowed channels improve the general performance of CNN in image tasks by strengthening gradient flow and parameter efficiency.

## 2.1.2 Segmentation Methods

Image segmentation has an important role in image understanding and has become dominated by deep learning methods. The CNN based segmentation models have outperformed more classical approaches in many biomedical scenarios. From magnetic resonance images, surgical images, to microscopy videos, these pixel-level models are designated to produce segmentation masks [35], uncertainty map [118], and reconstructed/synthesised images [126]. The architectures used for image segmentation can generally be categorised into a genre of models, whose input and output can be 2D, 3D, 4D with time axis, or even stacking with other layers of materials.

### 2.1.2.1 Fully Convolutional Network and Skip Connections

A seminal work under this question is introduction of Fully Convolutional Networks (FCNs, 2015) [83] whose architecture is exhibited in Fig. 2.1. The backbones of original FCN tested in the paper include AlexNet (2012) [64], VGG (2014) [106], GoogLeNet (2015) [110]. It has 'fully convolutional' in its name because it substitutes all the fully-connected layers(FN) in the backbones with 1x1 convolutional layers. As a solution to the problem of scale, a trainable up-sampling layer is employed to map the coarse features to the finer original size. To make the segmentation prediction more precise and benefit a model's gradient flow, [83] suggested using a trick of combining the deep up-sampled maps with the early high-resolution features. This trick has a profound effect on many later works. One contribution of FCN is it shows the feasibility of training a deep convolutional structure of full-size projection in an end-to-end manner. Different from the encoder-decoder design, the model has an asymmetric shape. Also, such architecture can be applied on the images of any size without taking additional parameters. It achieves 62.2 mean-IoU on VOC2012 test set with 20% relative improvement compared to the state-of-the-art at the time with 134M parameters.

In Fig. 2.2, some examples generated by FCN are displayed. It is not hard to find that some areas, such as the bicycle wheel in the first picture, the rider's leg, and the saddle in the second picture, are not well processed by the model. These revealed a

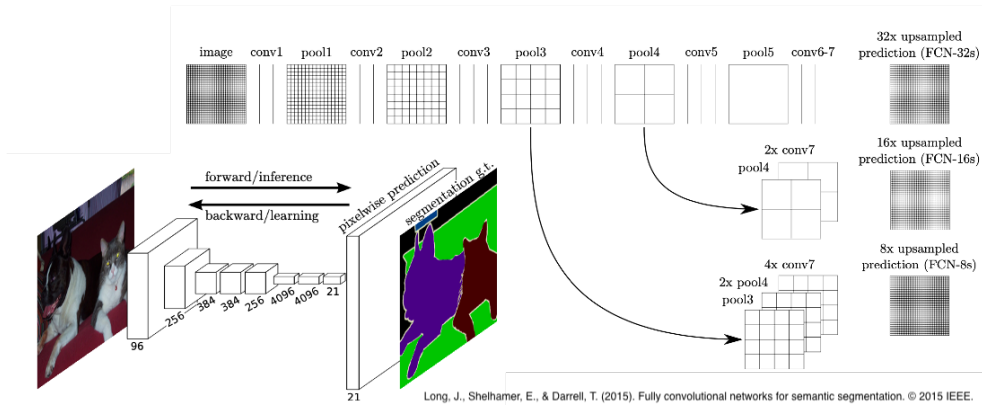


Figure 2.1: The structure of a typical fully convolutional network [83]. It has three basic components, activated convolutional layers, pooling layers, and a 'deconvolution' (transposed convolution) layer at the end. The network firstly downsamples an input image in steps to summarise local context information, then up-scales the last feature map back to the input size. For vanilla FCN, unlike from those that followed, it used a rather large kernel, e.g.  $14 \times 14$ , and increased the receptive field of nodes through pooling and setting kernel at particular depths with great stride.

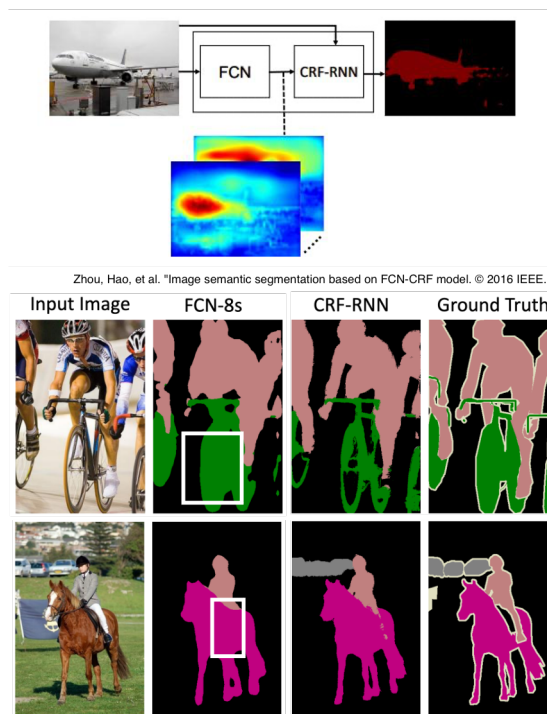


Figure 2.2: This image demonstrated the issue that CRF-FCN going to solve. The work uses conditional random field to pass local semantic information and improve the ability in detecting object boundaries. The output of CRF-FCN can produce a more precise segmentation.

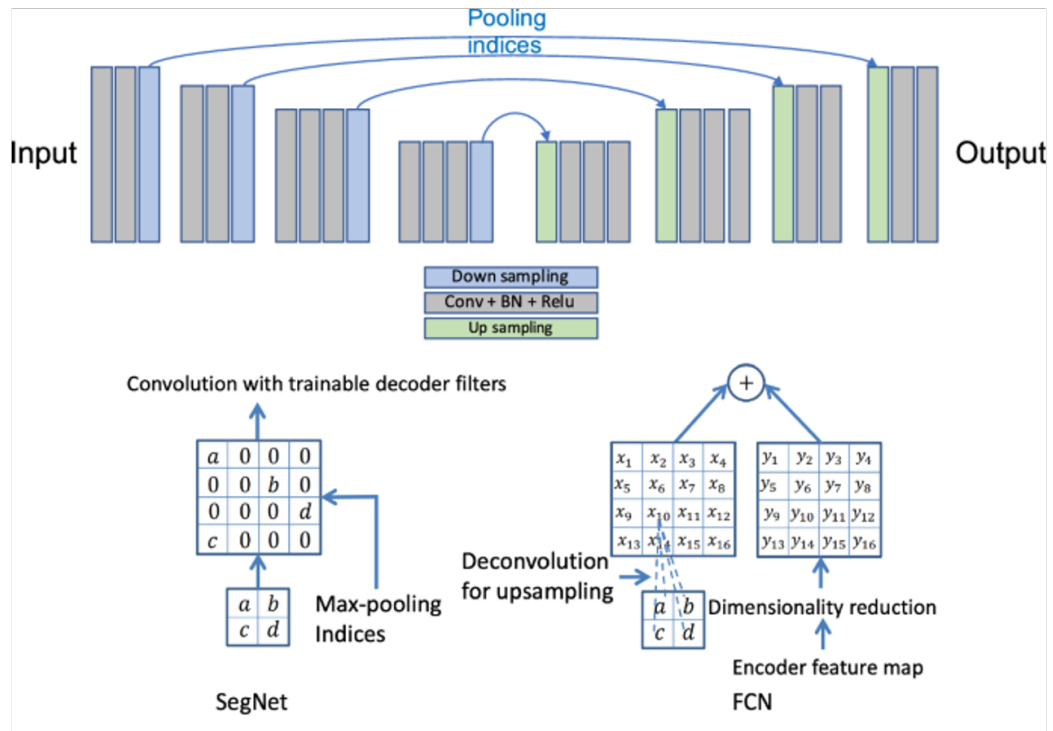
problem of the algorithm whose receptive field is too large before upsampling. The impact from a bottleneck layer pixel is placed indiscriminately onto a large area of the output. This could result in a coarse segmentation.

This issue was noticed by [144] who suggested a solution which involved combining a Conditional Random Field (CRF) into the workflow of the FCN. The work used a recurrent module to simulate the process of solving CRF, and added the module after a CFN. Thus the entire structure can be updated during back-propagation. Its energy function was designed as Eq.2.2 shown, in which the unary energy components  $\Psi_u(\mathbf{x}_i)$  is estimated by a FCN and the pairwise energy components  $\Psi_p(\mathbf{x}_i, \mathbf{x}_j)$  measure the relative energy from nearby pairs of pixels.

$$E(\mathbf{x}) = \sum_i \Psi_u(\mathbf{x}_i) + \sum_E \Psi_p(\mathbf{x}_i, \mathbf{x}_j) \quad (2.2)$$

The approaches have demonstrated the capability of a CNN model in pixel-level classification tasks. How to improve the architecture to make middle layers more expressive and address the upsampling limitations are the two open questions left to FCNs. SegNet [4] introduced a balanced process. As shown in Fig. 2.3. Instead of spending most parameters on the feature downsampling stage, SegNet suggests a symmetric structure, as known as the encoder-decoder style. This makes the features have a gradual evolution from lower-resolution to higher resolution. Also, it uses a different upsample operator which stores the pooling indices generated in the encoding stage as upsampling reference for decoding [94]. These index maps carry context information. Compared to transposed convolution, it is parameter-free and can bring more variations to an upsampled map if it has a convolutional layer after it. The SegNet shows an elegant way of integrating index map information into forward propagation. Our model, however, will leverage this problem from the angle of indexing features.

The transposed convolution was initially proposed by [94] in 2015. The weights in a transposed convolution kernel are trainable but do not mean such a function can perform a complex transformation after training. Its procedure can be interpreted as unfolding a repetitive pattern and then adding up the overlapping pixels. Considering the modelling efficiency, bi-linear interpolation is also very efficient, and potentially can wrap more context information during upsampling. A combined unit consisting of convolution and interpolation are used often in recent works such as the state-of-the-art [62].



Copyright, Asgari Taghanaki, Saeid, et al. Deep semantic segmentation of natural and medical images: a review. Springer, 2021.

Figure 2.3: The architecture of SegNet uses a symmetric encoder-decoder structure and Conv+bn+ReLU basic unit. Index information is passed from encoder to decoder used as upsampling reference. A visualised comparison between index guided upsampling (SegNet) and transposed upsampling (FCN).

### 2.1.2.2 Symmetric Networks

For semantic segmentation, the most frequently used method today is the U-Net (2015) [101] and its variants such as the self-configuring nnU-Net [59] which is the current state-of-the-art. The model's principle structure is illustrated in Fig. 2.4. It builds on the merits of previous works, and is fully convolutional. U-Net type models usually have a symmetric structure and learnable transposed upsampling nodes, in which shallow features are bridged to the deep features having abstract information mingled with texture signals. Developed on the original U-Net, new models continue to improve performance. Some important variants such as dense connection [28, 74], dilated convolution [15], and residual blocks [72, 141] have emerged recently. V-Net [88] expands the model to voxel-level on MRI data and suggested using Dice coefficient to address the server imbalance between foreground and background. Other attempts such as [44] made the network trained on CT Images progressively more dense.

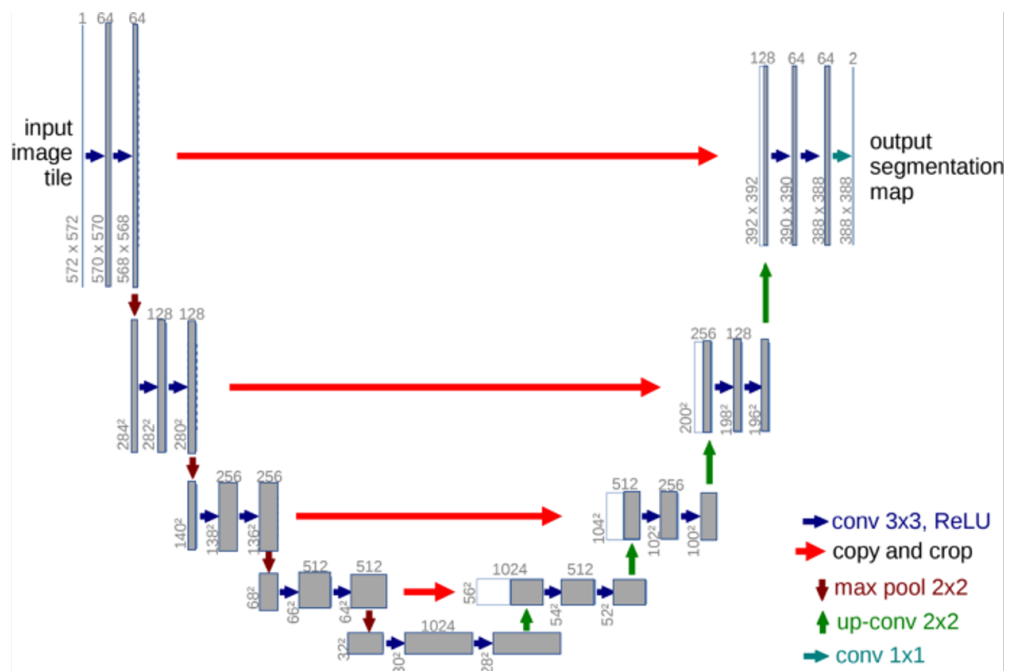
The core of U-Net is how the model handles coarse and fine features, where the bridge connections link intermediate stages of decoder and encode enabling more gradients to reach earlier layers during training. A 'residual-like' effect is happening during this process. The encoder path can be regarded as a process of refinement, It catches essential information on a coarse resolution. In the other words, some subtle details are lost during encoding. And the feature bridge gives a second chance to retrieve this information back during decoding. With this information mixture, a U-Net model is better in both locating local edges and locating large object areas comparing to SegNet.

As discussed in [142], the empirical receptive field is usually found smaller than its theoretical range. It increases linearly when a network goes deeper and is also influenced by the other factors such as kernel size, step stride and pooling ratio. It is critical to find a proper receptive field scale to avoid missing neither local details nor long range information dependencies. To achieve this with the scale of output maintained, some modules such as dilated convolution [17, 18, 136] and pyramid representation [1] were designed.

### 2.1.2.3 Dilated Convolution

From the aspect of efficiency, dilated convolution is favoured by the tasks with limited computational resources [97]. This can be realised from its three characteristics,





Copyright, Asgari Taghanaki, Saeid, et al. Deep semantic segmentation of natural and medical images: a review. Springer, 2021.

Figure 2.4: Earlier than SegNet, U-Net re-designed the classification-oriented CNN model to a symmetric encoder-decoder style. Features are analysed on five scales. The early feature maps are concatenated with the high-level decoded information in U-Net, and then the combined node is compressed by a  $1 \times 1$  conv node. Blocks are utilised as the basic units in model. For U-Net variants, the model shape is generally replicated and most innovation focuses on block design.

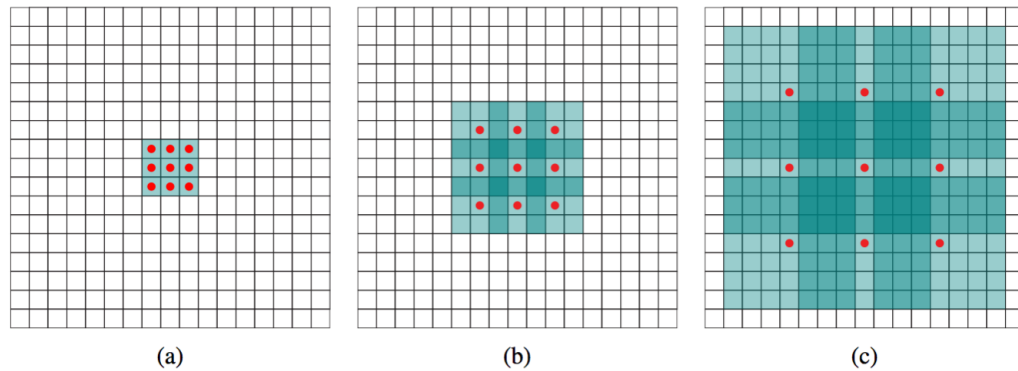


Figure 2.5: Dilated convolution [136] can increase receptive field exponentially with no additional parameters. The  $\mathbf{x}_a$  is a normal feature map where each position covers a  $3 \times 3$  receptive field. By applying dilated convolution, the area for the central position in  $\mathbf{x}_b$  is broadened to  $7 \times 7$ . After two rounds, a kernel on  $\mathbf{x}_c$  is extracting information from a  $15 \times 15$  equivalent field in  $\mathbf{x}_a$ .

stated as no extra parameters, no additional computation and the original image size is retained [17]. In this context, a normal convolutional kernel is a dilated kernel with dilation degree equal to 1. The dilated degree here means the sampling position gap, as shown in Fig. 2.5. A dilated convolutional node of  $(k_{size} = 3, k_{dilate} = 2)$  can cover a receptive field as large as  $5 \times 5$  from its previous layer. By stacking dilated convolutional layers, the receptive field of a model grows exponentially with the number of layers.

In the case where semantic hotspots are sparsely distributed, a dilated kernel can summarise context information. Comparing to pooling, it better maintains a feature's local spatial information. As a dilated layer with stride is a potential replacement of pooling layer, it saves the memory cost for downsampling. However, there are problems with the method. Aggressive use can cause a feature gridding effect on feature maps. In this case, the information of neighbour positions can be discontinuous after several rounds of dilation. Also, as its spatial skipping is not intensity guided, detecting a pattern in different scales is not the intention of the method.

Huang et al. [119] explored the policies for using dilated convolution. They found that, in order to avoid the problems, a kernel should meet three requirements.

1. The dilation degrees from different layers should not have a greatest common divisor that is greater than 1. For instance, a degree set  $r = \{2, 4, 6\}$  would lead to a gridding effect.

2. It is better to have a repetitive dilation pattern, such as  $r = \{1, 2, 3, 1, 2, 3\}$ .
3. In order to make a series convolution cover any hole and edge, the maximum distance  $M$  between two nonzero values should meet  $M \geq K$ ,  $K$  is the kernel size,  $r_i$  represents the dilation degree of layer  $i$ , and  $M$  is defined by,

$$M_i = \max(M_{i+1} - 2r_i, M_{i+1} - 2(M_{i+1} - r_i), r_i) \quad (2.3)$$

If there are  $n$  layers, the default value is set to  $M_n = r_n$ .

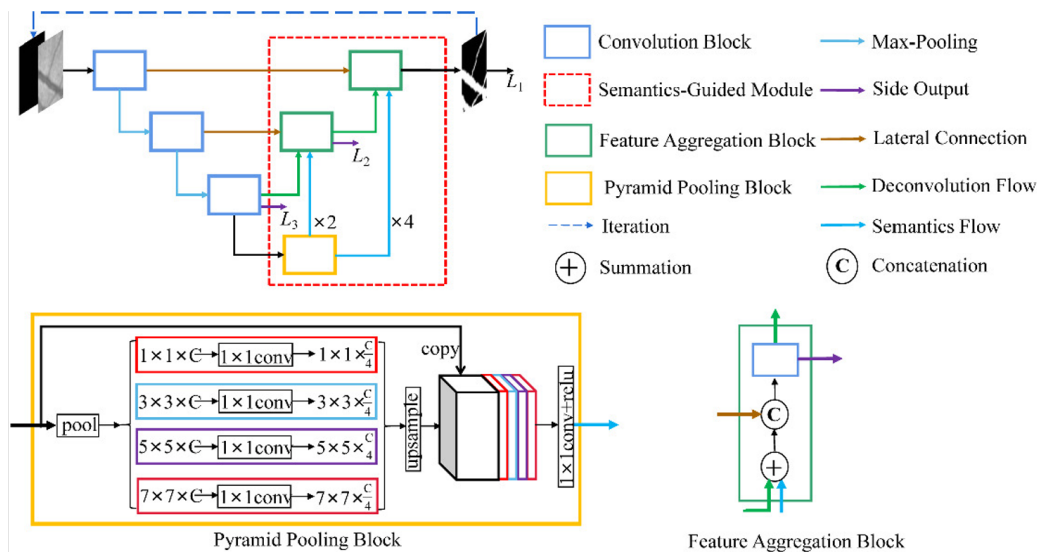
#### 2.1.2.4 Pyramid Representations

Boosting context involvement from another angle, the method of pyramid representation [1] has a long history which can be found in early conventional image analysis techniques and is still found very helpful in recent state-of-the-art studies. In deeplab v3 [18], both methods are integrated into the model. The application of pyramid representation can be found in the tasks of image classification [38], object detection [76], and segmentation [45]. The idea behind it is simple. The input feature maps are spatially down-sampled with different ratios, then analysis nodes are applied in a parallel manner on them and the extracted features are up-sampled back and summarised. It takes linearly more computation when the analysis kernels are sharing the weights, and needs linearly more parameters when the kernels are independent. Usually, a pyramid representation is believed to be a better choice when the necessary information for a decision are sparsely distributed over the entire feature map.

Its cost is not decided by the search scope but the number of ratios. A pyramid representation module is a more a scale search module. Other approaches such as adding more layers, use larger kernel sizes, and increasing the dilation degree can be relatively more inefficient in this case. However, as several copies of feature maps are held simultaneously, the method involves greater memory consumption.

#### 2.1.2.5 Boosting Local Segmentation

The challenge of axon segmentation in this project is mostly about generating a precise boundary. As the curvilinear objects are thin and long, a broken trace rendered by false negative segmentation would deliver a drastically different message and cause an



Copyright, Asgari Taghanaki, Saeid, et al. Deep semantic segmentation of natural and medical images: a review. Springer, 2021.

Figure 2.6: The structure of a recursive semantic-guidance network, where the core module is enclosed by the red dash square. The model is comparatively small and runs recursively. The vessel information is distilled and the area average intensity is strengthened. During training, higher weights are assigned to the later iterations. The model uses a pyramid pooling module to handle the objects with varying thickness. The success of recursive semantic guidance also confirms that it is practical to apply multiple curvilinear semantic detectors in a row for vessel-like patterns. This strategy is adopted in the MASC experiments.

inaccurate conclusion. Most useful information for a neural object is enclosed within a tight area making local segmentation a critical requirement in axon modelling.

In the literature of vessel research, similar issues exist. For example, the task of retina vessel segmentation, where small vessels are often missed, and uneven brightness can cause optical tissue to be over-segmented. As the connection of these curvilinear structures will be studied, it is important to locate even very thin connections keeping the topological structure complete. In [129], Xiao et al. proposed a weighted attention module. Combined with a U-Net model, it enables the processing to focus more on vascular patterns.

The module of recursive semantic-guidance was suggested by [134], in which the local information extracted from pooling representations are aggregated. Information

is distilled during several runs of forward-propagation. Every run is treated as a refinement step. A similar repetitive use of an encoder-decoder can also be found in an earlier work of post estimation [93], though the weights are not shared in it. In [134] Xu et al. claimed, with Lipschitz continuity assumed, such refinement can be conducted iteratively for a better segmentation result. This was also mentioned in [16,90]. The module needs no extra parameters for later iterations as kernel weights are shared.

Boosting local pattern precision is also explored by [16] and [53]. [16] introduced a method called second-order pooling. As Joao et al. introduced in 2012, the two second-order pooling functions, 2AvgP and 2MaxP are written as

$$G_{\text{avg}}(\mathbf{X}) = \frac{1}{n} \sum_{\mathbf{x}_i \in \mathbf{X}} \mathbf{x}_i \otimes \mathbf{x}_i^T \quad (2.4)$$

$$G_{\text{max}}(\mathbf{X}) = \max_{\mathbf{x}_i \in \mathbf{X}} \mathbf{x}_i \otimes \mathbf{x}_i^T \quad (2.5)$$

Unlike the common first-order pooling methods (e.g. 2D average pooling), the second-order pooling pays attention to the interactive information between the channel pairs generated by a local descriptor such as a convolutional window. A similar idea published recently is the local relation network [53], which usually is closely related to the Vision Transformer (ViT) [31] family. The idea behind the local relation is as,

$$F(\mathbf{p}', \mathbf{p}) = \text{softmax}(\Phi(f_{\theta_q}(\mathbf{x}_{p'}), f_{\theta_k}(\mathbf{x}_p)) + f_{\theta_g}(\mathbf{p} - \mathbf{p}')) \quad (2.6)$$

$$\Phi(\mathbf{q}_{p'}, \mathbf{k}_p) = -(\mathbf{q}_{p'} - \mathbf{k}_p)^2 \quad (2.7)$$

$$\Phi(\mathbf{q}_{p'}, \mathbf{k}_p) = -|\mathbf{q}_{p'} - \mathbf{k}_p| \quad (2.8)$$

$$\Phi(\mathbf{q}_{p'}, \mathbf{k}_p) = \mathbf{q}_{p'} \cdot \mathbf{k}_p \quad (2.9)$$

Rather than executing outer products on channel directions, small patches are sampled from every position using a sliding window and the calculation is conducted on spatial dimensions. Similar to an attention unit, the inputs are asymmetric, encoded by  $f_{\theta_q}$  and  $f_{\theta_k}$  respectively. Here, the context information is local with the relative co-ordinations integrated at the same time.

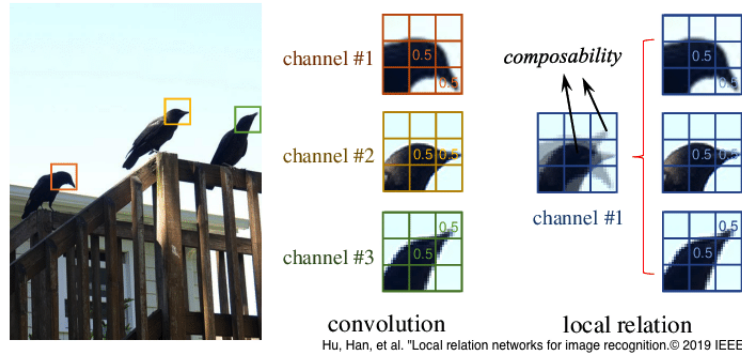


Figure 2.7: The comparison between a 3-channel  $3 \times 3$  conv layer and a single-channel  $3 \times 3$  local relation layer. A local relation layer can overcome the position variations within its effective area when the convolution method needs to model the cases respectively.

Good performance was observed comparing to the state-of-the-art at the time. Beside the conventional correlation function, squared difference (Eq.2.7) and absolute difference (Eq.2.8) are also tested in the experiments [53]. The results show their mean top-1 accuracies on ImageNet-1k are very close.

### 2.1.2.6 Instance Segmentation

Developed from semantic segmentation, instance segmentation refers to the models whose outputs contain differentiated object individuals (with rough or precise boundaries). Most instance segmentation models are from one of the following categories;

1. proposal + classification + segmentation (semantic knowledge),
2. segmentation + clustering,
3. dense proposal + segmentation (individual knowledge).

A modern milestone of instance segmentation was introduced in [43]. Its strategy was soon improved, in which mask proposals are generated and followed by classification [114]. The most well-known architecture is RCNN. Due to its multi-stage processing, it is usually difficult and time-consuming to train such a model.

Fast RCNN [41] and Faster RCNN [100] improved the structure, by characterising detection windows and combining the feature extraction steps. Mask RCNN [47] is designed based on Faster RCNN, adding a segmentation layer after the detected box to provide a masked shape.

Another method of instance segmentation is to apply pixel clustering on the semantic segmentation outputs [5]. The clustering step can be deep feature based or completed by a shape-relevant algorithm (e.g. watershed). This method is very popular in biological image processing [5], especially the pathohistology task of segmenting cells and cell nuclei [63].

Another kind of model that is becoming popular is called the dense sliding window method [19], in which spatial information is densely sampled, and a pyramid process of analysing feature maps is taken.

## 2.2 Matched Filters

Matched filtering was widely applied in radar pulsed signal detection [6, 83], and then be extended to 2D image analysis. The method was found very effective given sufficient prior knowledge of target pattern [31], but also relies on good noise control [149]. The typical form of this technique uses a sliding pattern filter to calculate the correlation with the sampled data in the window. The response score can be expressed as,

$$g(n) = \sum_{i=1}^k f(n+i)w_i \quad (2.10)$$

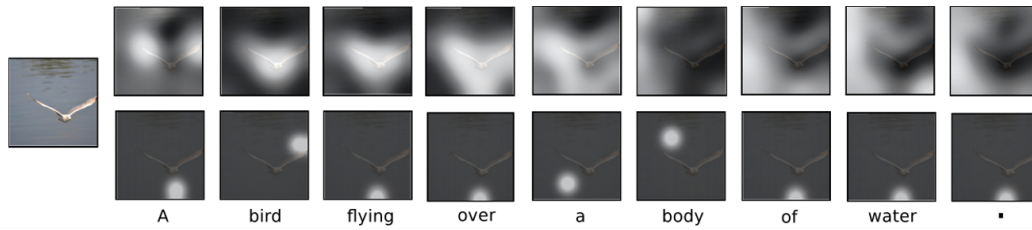
where  $f(i)$  is the signal function,  $n$  is the selected position of the signal function,  $k$  is the length of the desired pattern, and  $w$  is the matched filter. Its 2D form is,

$$g(x,y) = \sum_{i=1}^w \sum_{j=1}^x f(x+i,y+j)h_{ij} \quad (2.11)$$

However, in practice, to boost the desire intensities and enhance the signal-noise ratio, the noise pattern also need to be considered when generating  $h$ ,

$$f_o(x,y) = \{f_i(x,y) + n_i(x,y)\} * h(x,y) \quad (2.12)$$

Matched filters are found especially useful in blood vessel detection and fingerprint processing [106, 110]. In most cases, the desired pattern can be relatively easily extracted when the noise function can be much harder to be addressed properly. This inspired us to combine the spirit of matched filters with modern deep learning methods and use gradient descent to learn the function.



Copyright, Xu, Kelvin, et al. Show, attend and tell: Neural image caption generation with visual attention, PMLR, 2015.

Figure 2.8: A visualization of attention weights from both 'soft'(top) and 'hard'(bottom) attention [132]. The weights maps are produced to generate the words below. Both methods generated the same caption in this test. And lighter color means higher weights.

## 2.3 Attention in Image Processing

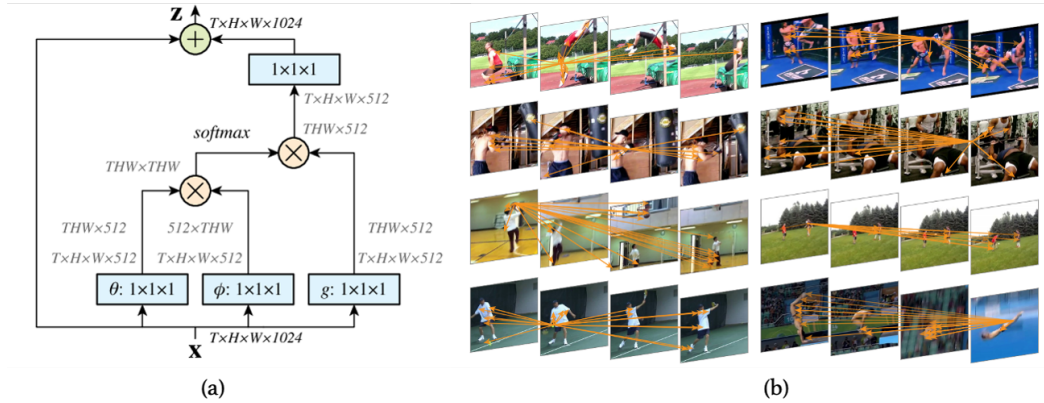
The attention mechanism in the modern sense was invented for nature language processing [115]. For example, in a sentence, attention weights are given to the positions according to their relevancy. A key word is more likely to get a higher weight thus a greater chance to impact the meaning of this sentence. To be more specific, an attention weight measures the semantic relation between a given position and a neighbour position.

Transplanting the algorithm to 2D data, the attention for images calculates the semantic domain distance between feature vectors. A sensible domain projection and matching leans on stable semantic meanings which makes pre-training/initialisation a critical factor in training a successful attention model.

In the work of Xu et al. [132], the attention mechanism is applied in an image caption generation task. In this paper, Xu et al. discriminate two types of attention, shown in Fig .2.8 as 'soft' and 'hard' attention. A deterministic 'soft' attention module learns weights from the entire image. The input feature map is multiplied by the attention map to create an attention embedded output. 'Soft' attention can be regarded as a weighted combination of input features. Under the limitation of  $sum = 1$ , a focused area will suppress the other irrelevant regions. For example, most contribution for generating caption 'bird' in Fig. 2.8 comes from the bird region, and the water part gets less attention.

A 'hard' attention, however, uses proposal maps to constrain its attention just as an instance segmentation model would do. As only a small patch in the image is sampled, its calculation is more efficient than the 'soft' scheme. Of course, the selection of focused patches is feature-based. A 'hard' attention is not completely differentiable





Wang, Xiaolong, et al. Non-local neural networks. © 2018 IEEE.

Figure 2.9: (a) In a non-local block, query and keys maps are encoded by different functions. The matching scores are rectified by a softmax node and then combined with the semantically encoded result  $g(\mathbf{x})$ . (b) Paired feature sources between consecutive frames extracted from videos.

as the proposal map is binary, and it generally requires additional functions to boost training.

Another important attention application in image analysis is the non-local neural networks [120]. Just like 'soft' attention models, a non-local net builds pairwise connections among all positions. Some transformer [115] flavoured non-local blocks substitute the vanilla attention units in the architecture. Its core function (Fig. 2.9(a)) can be described by,

$$y_i = \frac{1}{C(\mathbf{x})} \sum f(\theta(\mathbf{x}_i), \phi(\mathbf{x}_j)) g(\mathbf{x}_j) \quad (2.13)$$

in which feature map  $\mathbf{x}$  has  $C$  channels, and  $i, j$  represents two positions on the map.  $C(\mathbf{x})$  is a normalisation factor. Feature  $\mathbf{x}_i$  and  $\mathbf{x}_j$  are encoded separately by function  $\theta$ ,  $\phi$ , and  $g$ . Used like query and key, a matching score is computed as their product. The score is activated by a softmax function before being passed to the semantic value map  $g(\mathbf{x}_j)$ . Similarly, the process can be described as a weighted average over  $g(\mathbf{x}_j)$ . Several different  $f(\mathbf{x}_j, \mathbf{x}_i)$  functions were tested in the paper, including Gaussian, embedded Gaussian, dot product, and concatenation followed by linear combination. Though the best-performed model in the paper uses a self-attention [115] styled function, the modelling performance was found not sensitive to a choice. A similar observation was also recorded in [53].

Some interesting experiments can be found in [73], where Li et al. showed that

other choices of function  $f(\mathbf{x}_i, \mathbf{x}_j)$  and  $g(\mathbf{x}_j)$  would not make fundamental different with the task of semantic segmentation. In this case, a similar effect can be preserved even making  $\mathbf{x}_i$  and  $\mathbf{x}_j$  interchangeable, by which the  $f(\mathbf{x}_i, \mathbf{x}_j)$  output would be a symmetric matrix.

Instead of only paying 'attention' to the labelled area, Reverse Attention Network (RAN) [57] studied the possibility of counting the opposite features to provide a cross attention effect. [71] leverages a pyramid representation with attention units to increase the receptive field encoded in each feature vector. OCNet [138] integrated spatial attention with convolution for capturing both global and local information, and reported remarkable accuracy on several datasets. The attention paths on spatial-wise and channel-wise were tried in [34] based on a dilated convolution backbone, and outstanding performance at the time was claimed.

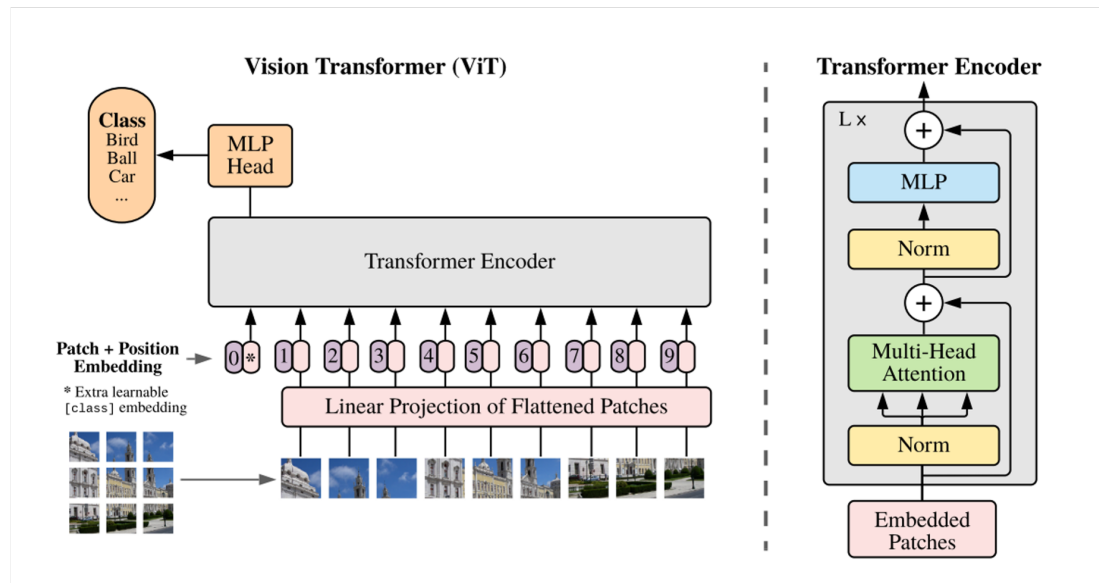
A problem of attention-using models is they are very expensive in both computation and memory consumption. Attempts were made on several aspects. For example, PSANet [143] simplified the encoding process (or key-query process in self-attention) by adding two consecutive weighted combination nodes on  $\mathbf{x}_j$  with relative position information embedded. The attention calculation is split into two steps, row and column, by CCNet. Interlaced sparse self-attention (ISA) [56] also uses two steps (long-range and short-range attention) to simplify the calculation.

Inspired by a similar motivation, EMANet [73] proposed a much simpler way. The model leverages an expectation maximisation module to move kernels to the centre of the clusters in the high-dimensional feature space. Though the model is, in theory, constrained by the limitations of the expectation maximisation method it takes (depending on the shape of the clusters), it provides a very straightforward way to interpret and construct attention.

### 2.3.1 Vision Transformers

Another popular genre of attention mechanism in vision is Vision Transformers (ViT) [31] [79]. These standardised the structure of a basic vision transformer block, and prove the viability of a transformer-only model. Unlike traditional vision attention, a ViT unit works on patches rather than pixels, and embeds positional information into the features. Using ViT can help mitigate the information loss during downsampling which is used to manage the pixel-level computation cost in vanilla attention.

As shown in Fig. 2.10, typically, such a block has three parts, i) layer normalization, ii) multi-head self-attention, and iii) multi-layer-perceptron based information



Copyright, Dosovitskiy, Alexey, et al. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

Figure 2.10: An overview of ViT model and its transformer block. The model splits an image into fixed-size patches and wraps them with position information. The position embedded features are then fed to a transformer block, in which they will be tokenised. A vision transformer block is made from a multi-head attention layer and a MLP information extractor. Features are normalised before and after the self-attention node in a block.

integrator. The input patch size can be large, for example in the work of Dosovitskiy et al. [31] images are split into 16x16 patches. Swim transformer [79] improved the cropping strategy by adding shift to patch merging. This provides the sampling with equivalent density and is computationally more efficient than using a common sliding window. The tokenised patch ensures a sufficient receptive field for extracting context information. The strategy has also been used in medical image segmentation [113, 140].

The ViT models gave people another choice other than convolution. In theory ViT can achieve equivalent accuracy with less computation complexity. But in reality, it is constrained by the present software and hardware support. Another issue suggested by [31] is the transformer models need more training samples to learn the key, query, and value conversions. In bio-medical scenarios, the images objects are usually simpler in structure and have fewer categories to deal with (comparing to the tasks like ImageNet-3K) and more local patterns (most features can be determined by local context rather than global context).

## 2.4 Rotational Symmetric Pattern

In this project our goal was to study the axon structure. An axon has a particular shape, which gives us the opportunity to optimise our algorithm to the task. Axon objects tend to be curvilinear structures which can occur at any orientation. In image processing, we already have steerable filters, a sophisticated method used in edge detection and texture analysis. Some of the early examples like [33], and recent ones with deep learning [40, 86] have shown that it is feasible to use a steerable strategy to solve this problem. Beyond that, we also extend the model with scale-invariant features.

### 2.4.1 Steerable CNN

The conventional strategy [33] in steerable construction is to use an atomic basis. By manipulating the basis combination weights, orientational equivalent filters can be created. In the paper of Steerable CNN [25], Cohen et al. defined a rotational transformation mechanism  $\pi_\theta$  as,

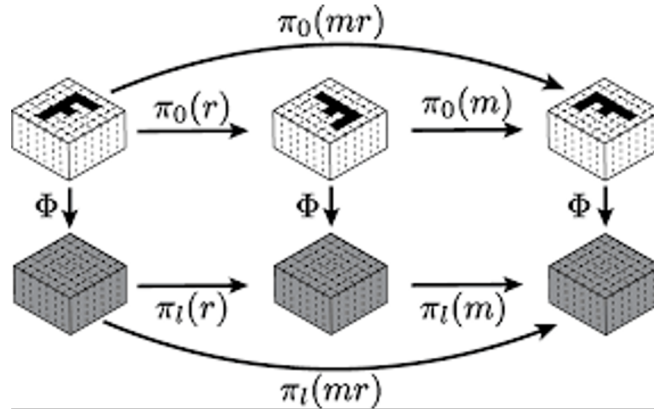
$$[\pi_\theta(g)f](\mathbf{x}) = f(g^{-1}\mathbf{x}) \quad (2.14)$$

$G$  is a group of rotation transformation, and  $g \in G$ . and  $f(\mathbf{x})$  is the feature signal. The pixel at  $g^{-1}\mathbf{x}$  is projected to the rotational-invariant space via operator  $g$ . The entire process is linear. Given a steerable representation  $\pi$  having  $\Psi\pi(g) = \pi'g\Psi$ , we must have the transformation  $\pi' \in G$ , thus

$$\pi'(gh)\Phi f = \Phi\pi(gh)f = \Phi\pi(g)\pi(f)f = \pi'(g)\Phi\pi(f)f = \pi'(g)\pi'(f)\Phi f \quad (2.15)$$

Here  $\Phi : \mathcal{F} \rightarrow \mathcal{F}'$  is a MLP that projects an input feature to another. More specifically, the rotational transformation  $\pi$  in steerable CNN [25] does not depend on the input space.

Though the method can be extended to continuous groups in theory, the authors mentioned it has potential discretisation and approximation issues. Thus the model is not ready for arbitrary rotation in practice.



Copyright, Cohen, Taco S., and Max Welling. Steerable cnns, 2016.

Figure 2.11: Demonstrate the transformation consistency of involving  $\Psi$  and  $\pi_0$ . The result is independent to the transformation path it takes.

## 2.4.2 G-CNNs

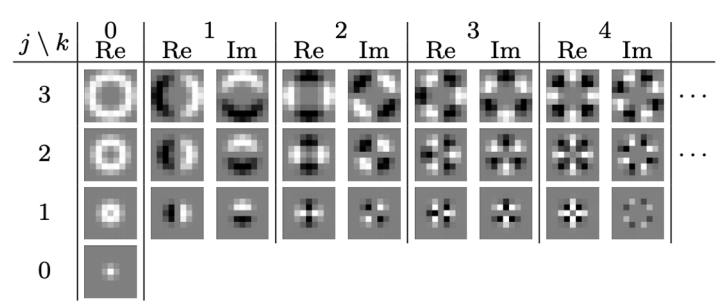
Closely related to the steerable CNN, Group CNN (G-CNN) [24] provides a more comprehensive theoretical framework. As a practice of linking group theory with neural network, the G-CNNs can produce a set of filters from an enclosed group, and these filters share the sensitivity to the corresponding homogenous patterns. [51] extend the original 4-fold ( $4 \times \frac{\pi}{4}$ ) rotational symmetry to 6-folds symmetry with hexagonal tilting. Another important work in this line is [8], which highlights the application of rotated filters in medical image processing, and suggests a method for constructing a rotational equivalent filter group that is exact and efficient on a grid basis. In [122], a more general framework of the steerable filters in neural networks is proposed.

## 2.4.3 Harmonic Basis Methods

Harmonic filters [125] [123] are another popular type of steerable filters. A circular harmonic basis expressed in polar coordinates has the form

$$W(r, \phi) = \mathcal{N}(r)e^{i(k\phi+\beta)} \quad (2.16)$$

In this equation,  $\mathcal{N}$  is a Gaussian function, and the second half is a unit complex function that enables the harmonic filters to be presented in any direction. The steerable filters in this case [124] are a weighted combination of the circular harmonic basis, however, such methods are usually expensive and hard to train. A similar strategy is



Weiler, M., Hamprecht, F. A., & Storath, M. Learning steerable filters for rotation equivariant cnns. © 2018 IEEE.

Figure 2.12: A harmonic function  $\psi$  proposed in [124], which is defined as  $\psi_{j,k}(r, \phi) = \tau_j(r)e^{ik\phi}$ . Its extension on radial direction is presented by rows, and the columns represent the angular frequency changes. The basis can be created in any size thus is able to conduct searching on multiple scales.

also presented by [112]. Larger kernels [40] that can contain more high frequency details are favoured but require exponentially more parameters.

Other hybrid methods such as [85] [81] leverage a Gabor generator to achieve a rotational-equivalent effect. Combined with convolutional kernel, the rotational detectors are adaptive to noisy patterns and were found to be helpful when processing texture-rich images. Though the methods use a strong assumption making them suitable for certain type of problems, these approaches have efficiency advantages in parameter usage comparing to the usual random seed pattern searching.

## 2.5 Modelling Structured Data

A good segmentation model can generate a precise mask for a target object. To analyse the morphology structure of an axon, the graphical data must be converted into structured format. A segmentation mistake can break a track or connect two axons, and fundamentally changing the topology. In this project, we observed large obstacle areas and complex object entanglement under microscopy. This makes instance-level segmentation problematic on axon data.

The object attributes will be studied on a frame by frame basis, which is a sequential modelling task. A tree structure is a natural choice for representing axons.

We have the demands of separating curvilinear(axon) objects, processing temporal sequences and analysing tree objects. All these tasks are about modelling structured

information. Here, we discuss the technologies that may be useful in solving our challenges. Specifically we will cover recurrent models and graph models.

### 2.5.1 Recurrent Models for Sequential Data

Recurrent Neural Networks (RNN) are a type of deep learning architecture for processing sequential data. Recently, transformers [115] [30] were found being able to generally outperform the conventional RNN models in many tasks. This was discussed the applications of attention and transformer in vision tasks in Section 2.3 and Section 2.3.1.

The RNN models still have some advantages compared to sequential transformers. Generally, they can work with smaller model sizes, are easier to train, and more robust when the context range is uncertain. Also, a transformer requires more training samples and are prone to overfitting [99] [11]. For bio-medical image analysis, where data can be hard to obtain, this factor cannot be ignored.

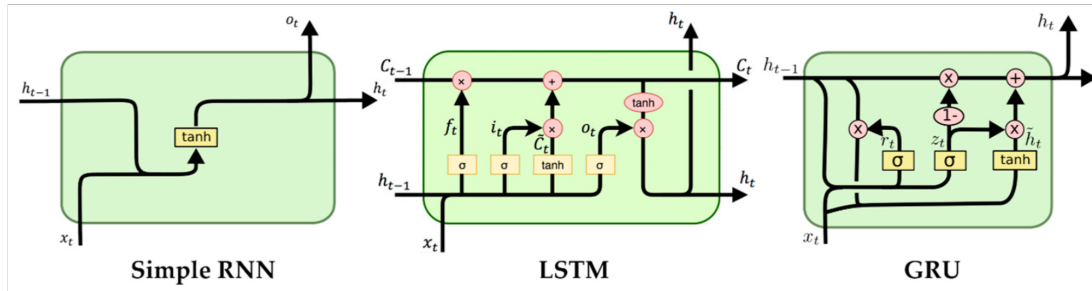
The classic RNN models have a problem of learning long-term dependency. This inspired another two types of models which are Long-Short-Term Memory (LSTM) [50] and Gated Recurrent Unit (GRU) [23]. These two concepts are also used in graph algorithms, which will be explained in the next section.

The process of LSTM can be interpreted as calculate cell state by combining a linear transform of the previous cell state and a linear transform of the encoded new input information. The two linear transformations here are learned separately based on the input and the hidden state from the last step. At the last, the new hidden state can be decoded from the updated cell state. Such structure suggested a set of learned weights to dynamically balance the stored information and the new information. With a non-linear activation function to manage the residual information (shown in Fig 2.13), gradients can flow smoothly through the steps, as the information accumulated effect is alleviated.

Gated Recurrent Unit [23] is another type of RNN block. Similar to LSTM, its core idea is balancing the information accumulation, which can be expressed as,

$$\begin{aligned} \mathbf{r}_t &= \sigma(\mathbf{w}_{rx}\mathbf{x}_t + \mathbf{w}_{rh}\mathbf{h}_{t-1} + b_r) \\ \mathbf{z}_t &= \sigma(\mathbf{w}_{zx}\mathbf{x}_t + \mathbf{w}_{zh}\mathbf{h}_{t-1} + b_z) \end{aligned} \quad (2.17)$$

$$\hat{\mathbf{h}}_t = \mathbf{w}_{hx}\mathbf{x}_t + \mathbf{w}_{hr}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + b_h \quad (2.18)$$



Copyright, dprogrammer.org, 2022

Figure 2.13: The data flow in RNN, LSTM, and GRU. In LSTM and GRU, information is controlled by gates to alleviate gradient explosion and vanishing. And this gives a recurrent model the ability of selective memory and allows it to remember more.

$$\mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \hat{\mathbf{h}}_t \quad (2.19)$$

The hidden state memory is controlled by  $\mathbf{z}_t$ . The replacement of preceding information is encouraged when  $\mathbf{z}_t$  approaches zero, thus new information can be remembered. Besides the classic LSTM and its simplified version GRU, other variants such as Quasi-Recurrent Neural Network [10] and Simple Recurrent Unit [69] made modifications to improve parallel processing, promoting the calculation to be more efficient.

## 2.6 Graph Neural Networks

A graph structure is constructed by a set of nodes and the connection between them. When analysing axons, an axon's structure can be converted to a polytree which is a special acyclic graph structure whose nodes are all connected and directed from a root node. Graph neural networks (GNN) empowered by perceptron and gradient back-propagation have been very effective in solving complex prediction and classification tasks on graphs. GNN models are employed in innovations in the area of feature embedding and image tokenisation. This combination has helped build applications such as protein interaction study [32, 58], and automatic fact verification on social media [82].

An image can be interpreted as a grid graph where every pixel is a node with 8 neighbour nodes (except boundary pixels). In that case, the interpretation of Convolutional Neural Network could be expressed as learning "shift-invariance" and "local



connectivity” [128] on an image graph. It suggests that GNN has the potential to become a way of unifying all kinds of neural networks.

In the axon project, for example, to improve robustness, a greedy strategy is taken where all potential axon breaks are merged with the main axon tree. Then we apply branch pruning which is a node classification task. The pruning decision model is graph based thus GNN is a reasonable choice here. In the other cases, GNN is used as a feature mixer [128] in the pre-processing stage of a sequence to sequence task.

### 2.6.1 Recurrent Graph Neural Networks

The modern form of Graph Neural Network was initially suggested by Recurrent Graph Neural Networks (RecGNNs) in [104]. Very close to the later Graph Convolutional Networks (GCN) [87] [61], the feature of a node in a graph is updated based on itself and the neighbour nodes. The information is exchanged and aggregated until an equilibrium status was reached. The hidden state at step  $t$  is defined as,

$$\mathbf{h}_i^t = \sum_{j \in N(i)} f(\mathbf{x}_i, \mathbf{x}_j, \mathbf{h}_i^{t-1}) \quad (2.20)$$

In this equation, function  $f$  is a message integration function,  $N(i)$  returns the neighbour nodes of node  $i$ . Developed from this basic form, sometimes, the edge information between the selected node  $i$  and its neighbour  $j$  is also included. One of the most popular works in this category is Gated Graph Neural Network (GGNN) [75], where a GRU module is borrowed to solve issues caused by the recurrent updating in the model. The information sources are alternatively aggregated as,

$$\mathbf{h}_i^t = GRU(\mathbf{h}_i^{t-1}, \sum_{j \in N(i)} \mathbf{W} \mathbf{h}_j^{t-1}) \quad (2.21)$$

The problems are also obvious. In RecGNN. all the nodes are required to be calculated in a recurrent manner, during which all the information generated at a step is held in the memory. When an input graph is large, the memory consumption would be immense. This problem was realised soon and a solution is proposed in [26], where node features are not updated simultaneously but following a selective strategy.

Similar to RNN styled models, a RecGNN usually follows an encoder-decoder approach. The extracted features will be projected to a task domain as necessary.

## 2.6.2 Graph Convolutional Neural Networks

Graph Convolutional Neural Networks (GCN) [61] [37] are the most popular form of GNN. Today, most GCNs can be classified into two categories [128] [145], as the spectral-based forms and the spatial-based forms.

### 2.6.2.1 Spectral-based method

The spectral-based models have a long history back to spectrum graph analysis. The method defines its operation in Fourier domain and leverages graph's Laplacian matrix to extract features, which is written as,

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \quad (2.22)$$

In this equation,  $\mathbf{D}$  is diagonal connection degree matrix,  $\mathbf{A}$  is adjacency matrix.

$$\mathbf{L}_{i,j} = \begin{cases} deg(i) & \text{if } i = j, \\ -1 & \text{if } i \neq j \text{ and } i, j \text{ connected,} \\ 0 & \text{otherwise.} \end{cases} \quad (2.23)$$

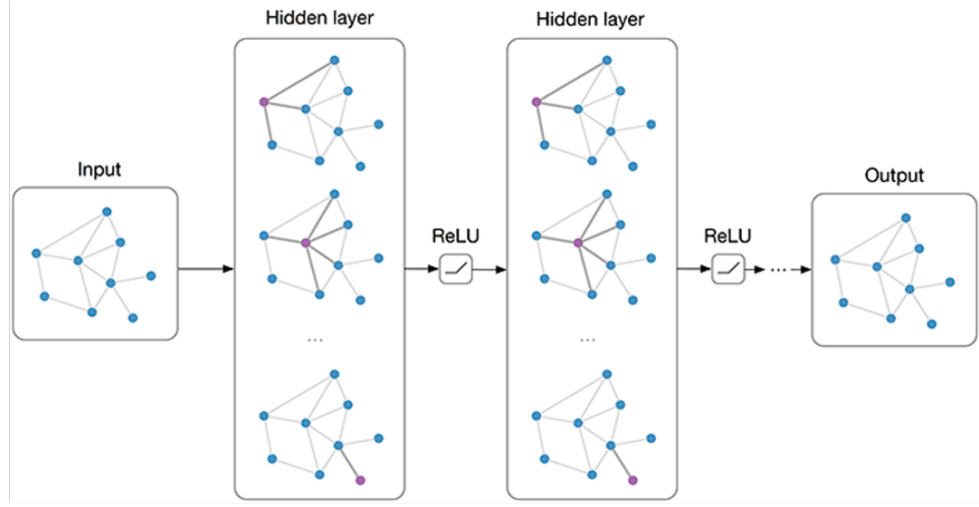
And, more often, its symmetric normalised form is used, defined as,

$$\mathbf{L}^{sys} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}} \quad (2.24)$$

The Laplacian matrix can depict the energy flow in a graph but requires the topological information of entire graph to be available. Thus the model cannot be directly applied to the scenario with many invisible nodes as well as node embedding tasks. The original spectral GCN cannot work on directed graphs [61]. In this framework, the strength of energy (information) flow on every edge is defined by the energy difference between the two connected nodes.

The spectral GCNs takes the eigenvectors  $\mathbf{U}$  of  $\mathbf{L}$  as a basis, projecting the node information into a new space. This process is usually named as the Fourier transformation on graphs. In Bruna et al.'s work [13], the node updating process is designed as,

$$\mathbf{X}^{t+1} = \sigma(\mathbf{U} \mathbf{g}_\theta \mathbf{U}^T \mathbf{X}) \quad (2.25)$$



Copyright, Nayak, Tapas. *Deep neural networks for relation extraction*. 2021.

Figure 2.14: The basic form of a spectral GCN. The model has a multi-layer structure. In each layer, information is collected from neighbour nodes, aggregated, and rectified. Node features are projected to the task domain before output.

where  $\sigma$  is a non-linear activation function,  $\mathbf{g}_\theta$  is a learnable parameter. The calculation aims to project the features to a basis space. After being processed by a linear operator, the features will be projected back and aggregated. However, this method is rather expensive, as its information aggregation is based on the entire graph.

In the work of Defferrard et al. [29], the method is improved where hyper-parameter  $k$  is used to define the receptive field for updating a node, and  $k \ll n$ . Also, a trick was used to simplify the decomposition calculation, which can be done directly on  $\mathbf{L}_j$ . Its simplified updating equation is written as,

$$\mathbf{x}^{t+1} = \sigma\left(\sum_{j=0}^{K-1} \mathbf{w}_j \mathbf{L}_j \mathbf{x}\right) \quad (2.26)$$

In [61], Kipf et al. further suggest to take  $k = 1$ , and turned the equation to,

$$\mathbf{X}^{t+1} = \phi(\bar{\mathbf{A}} \mathbf{X}^t \mathbf{W}) \quad (2.27)$$

$$\bar{\mathbf{A}} = \mathbf{I} + \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$$

where  $\phi$  is a non-linear activation function.

The spectral GCN is based on a graph Fourier transformation which is conducted over the entire graph. This transductive character limits the generalisation capacity of a spectral-based GCN model, as the basis learned from a graph cannot be converted to another graph with a different shape. Besides, the spectral method gives a solution to summarise information from the neighbour nodes of different distances but cannot differentiate the contribution weights from the nodes having the same distance.

### 2.6.2.2 Spatial based method

Spatial-based GNN is more straightforward. Following the idea of RecGNN, it extracts information from a defined range of neighbours and convolves. The central node updates its status after message aggregation. The information propagation for a spatial GNN is along the direction of edges which is specified in the input.

One of the classic models in this genre is NN4G [87]. It has two steps, an aggregation step and a readout step. In the aggregation step, the hidden states of nodes are calculated simply as the sum of the linear transformed hidden states of neighbour nodes, which is expressed as,

$$\mathbf{h}_i^k = \phi(\mathbf{w}_k \mathbf{x}_i + \sum_{j \in N(i)} \Theta_k \mathbf{h}_j^k) \quad (2.28)$$

It takes an unnormalised adjacency matrix which potentially cause completely different feature scales between nodes. In the readout stage, the representation of a graph is summarised by the weighted average of the mean hidden state of all nodes.

$$y = \phi\left(\frac{1}{k} \sum_{i=1}^n \sum_{t=1}^k \mathbf{w}_i \mathbf{h}_i^t\right) \quad (2.29)$$

MoNet [89] tried to design a universal framework for graph convolutional networks, and claimed that even the original GCN [61] (spectrum based) can be included as a special case on spatial GNN. It proposed a method to encode the relative position of a pair of nodes, the encoded relative position can be used to determine an adaptive weight on each neighbour node in the information aggregation stage.

Another work that enables adaptive weight in aggregation is Graph Attention Networks (GAT) [116]. Its weight measurement is not predefined but learned. Using an attention mechanism, a GAT model measures the pair-wise relationship of nodes by,

$$\mathbf{d}_{ij} = \text{softmax}(\phi(a[\mathbf{W}_1\mathbf{h}_i||\mathbf{W}_2\mathbf{h}_j])) \quad (2.30)$$

$$\mathbf{h}_i^t = \sigma\left(\sum_{j \in N(i)} \mathbf{d}_{ij} \mathbf{W}_3^t\right)$$

where  $a$  is a shared attention mechanism,  $\phi$  is LeakyReLU activation function, and  $[\mathbf{v}_1||\mathbf{v}_2]$  represents vector concatenation. The hidden states are updated as shown in Eq.2.30.

For tasks such as analysis of point clouds or knowledge graphs, the scale of the input can be very large, and the connections of a node can vary from one to thousands. To cope with this problem, GraphSAGE [42] leverages a neighbour sampling strategy, and puts the focus onto the aggregation function which is order invariant, such as mean, max, or an LSTM with order-shuffle input. GraphSAGE is often used in node embedding to generate positional estimation because of its inductive character.

Another interesting algorithm is Graph Isomorphism Network (GIN) [133]. The paper listed several conditions that the previous approaches cannot deal with and states the maximally powerful GNN is injective with multi-set neighbour features.

The injective aggregation and readout functions proposed by GIN are,

$$\mathbf{h}_i^t = \text{MLP}((1 + \varepsilon^t)\mathbf{h}_i^{t-1} + \sum_{j \in N(i)} \mathbf{h}_j^{t-1}) \quad (2.31)$$

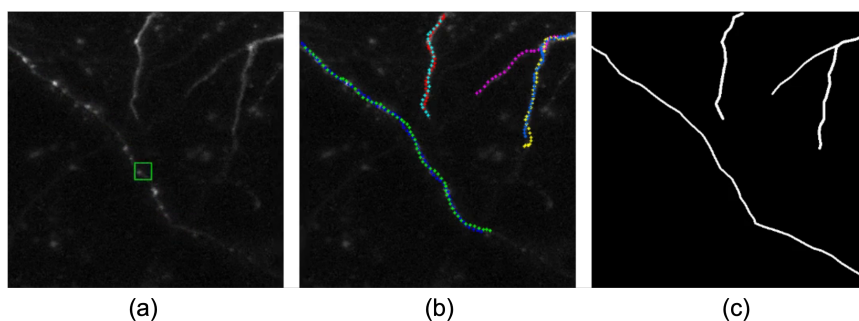
$$\mathbf{H} = [\mathbf{h}_1||\mathbf{h}_2...||\mathbf{h}_i...||\mathbf{h}_n], i \in G \quad (2.32)$$

$$\mathbf{h}_i = \sum_{t=1}^k \mathbf{h}_i^t \quad (2.33)$$

where MLP represents a multi-layer linear function, the graph representation  $\mathbf{h}_G$  is the final states concatenation of all nodes, and the final state of an individual node is the sum of its step states.

### 2.6.3 Axon Tracking Techniques

Tracing is one of the core tasks in this system. In the literature, semi-automated and automated methods were proposed for solving axon tracking problems on 2D images. It usually faces two challenges, locating axon pieces and ensuring the found axon pieces are complete.



Copyright, Deep Reinforcement Learning for Subpixel Neural Tracking Proceedings of Machine Learning Research 102:130–150, 2019

Figure 2.15: Reinforcement tracing makes decisions according to the patches, indicated in (a), sampled from a neighbor position. It starts from a pre-determined position. Once a decision is made, the same process will be taken again, step by step like the dots in (b) until the stopping condition is reached. The connection of these steps in (c) represents an axon trace, one axon trace each run.

A major group of methods leverages the elongated shape of axon fibre. The particle filtering approach suggested in [137] searches the image area guided by the generic path – only a limited number of pixels will be checked. The approach assumes an axon trace would not have significant direction change, which reduces the robustness of this model in more general conditions. Snake models [60] were used in contour and edge detection which is similar to axon detection. A model developed by Cai et al. [14] uses the repulsive power realised from context to determine the position of axons. But this could fail in axon entangled areas. Some other works use a spline to represent axons [3]. This improves a model’s interpretability, as axon branching order and geometry relationships can be quantified. The objects represented by a spline model also reserve their morphological structure. These results can be easily converted to tree structure which inspired us when designing the axon tree model in Chapter 5. The nature of spline makes it difficult to perfectly match a trace compared to the latest pixel methods.

Another group of methods follows the principle of tracking-by-detection. To enhance the modelling accuracy in noisy videos, Bowler et al. [9] includes both spatial and temporal information in feature vectors. This helps to overcome the image problems that occur on a single frame easily. Its features are cleaned by PCA and split into binary categories by a probabilistic classifier on pixel level. To improve a model’s calculation efficiency, classification can also be conducted on blocks or branches. Zhu et al. [147] sorts axons into blocks and then use Markov Random Field(MRF) to label them. A more advanced version of this uses Reinforcement Learning (RL) [146] to

make step decisions. In this case, tracing usually starts from a known position of axon and expands from it. A reinforcement learning action can be made in either a discrete (pixel) or continuous (coordinates) manner to extend the path with nearby candidate positions until the entire trace was found. However, local information is not sufficient for locating axon in large and blurred areas. Dai et al. [27] (Fig. 2.15) takes a 2-channel window of  $11 \times 11$  and  $22 \times 22$  pixels to absorb both local and slightly longer-range information (depending on resolution) as model input. Such an RL model for axon tracing requires a good reward function that is subjected to image conditions. For example, the work of Dai et al. [27] penalises large trace direction changes. But this may not always be correct, such as Fig .1.2. Agent-stopping rules are also critical as a model has to balance accuracy with efficiency and avoid over-segmentation when connecting the axon pieces blocked by various obstacles and low-contrast issues. These factors make it difficult for most RL models to accommodate unseen axon data.

# Chapter 3

## Multi Angle and Scale Convolution

### 3.1 Introduction

To address the segmentation of curvilinear and other isotropic patterns, we propose a novel semantic segmentation solution called Multi Angle and Scale Convolution (MASC). Inspired by hybrid convolution [85] [81], the method takes its kernel as a Hadamard product of free convolutional weights and constrained Gabor convolutional weights.

As shown in Fig. 3.1, isotropic patterns are common nature as well as in medical images. From aerial photos to MR and CT scans, the same textures can appear at a range of orientations and scales, but most present architectures treat them as different patterns. Moreover, isotropic property not only found in raw images but also in high-level features. Commonly processed by dozens of layers of sampling kernels and pooling windows, the positional information has been disturbed and the features are re-arranged. The same semantic message may be presented in several orientations. A good practice is to find these equivalent patterns on deep maps as mentioned in [92]. These facts assure the utility of a MASC unit which makes the convolution process rotation and scale invariant.

Updating directional kernels in balance is an implicit requirement of training a successful rotatable model. For a dataset with limited training samples which concentrating on only a few directions, the user should either use data augmentation methods to adjust the distribution or weight the orientation searching range. If required, a model should allow the user to constrain the orientational searching to a preferred smaller range. In MASC, it is feasible to precisely control its searching range depending on the prior knowledge, either decreasing or extending it. More details are introduced in



the Chapter 5.

We chose curvilinear tasks to demonstrate the idea for two reasons. First, axon objects are generally curvilinear and isotropic. There are several public datasets of retinal vessels which have similar properties and have been used to evaluate many other algorithms. Second, as the shape of the target object is relatively simple it is a good example to demonstrate the algorithm and the motivations behind it.

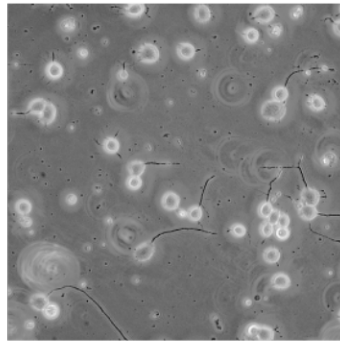
An idea solution should be able to address two critical challenges left by the other rotatable models and segmentation applications.

First, it should be fully differentiable. A MASC learns to build directional kernels from examples thus can achieve invariant detection adaptively. The previous kernel rotation solutions mostly require pre-constructed bases which are either static [25] [24] or cannot be generalize to arbitrary angle. The rotation module in MASC, also called Multi-Angle Convolution unit (MAC unit), leverages a kernel reuse scheme together with a novel message integration design called response shaping (see Section.3.2.4) to maintain a consistent set of directional kernels. The kernel rotation is justified by demonstrating the intermediate output and the covariance matrix of directional kernels in Fig. 3.11 and Fig. 3.7.

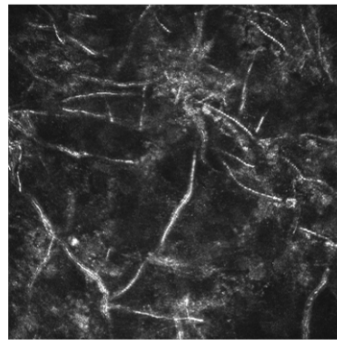
The second challenge is how to reduce the number of parameters in the model. The benchmark models such as R2U-Net(1.037M parameters), U-Net(2.6M/30M parameters) are initialized randomly. Such a greedy search routine has been found relating to intrinsic kernel redundancy [22]. With less than 0.4% (3,799 parameters) parameters and 12.5% training samples, a MASC model is a more economy choice which has equivalent/better performance on the dataset of CHASE-DB1 [96] and DRIVE [109] compared to the benchmarks. It has other desirable features including fast converging and good interpretability. More figures about these topics as well as the ablation studies are in Section.3.4.

This chapter will focus on the basic idea of MASC with its application on curvilinear patterns. An extended version generalised for more common objects is in the next chapter.

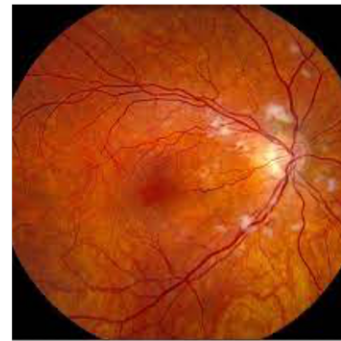
The design of MASC can be summarised by 5 procedures, and the relationships of them are visualised in Fig.3.2.



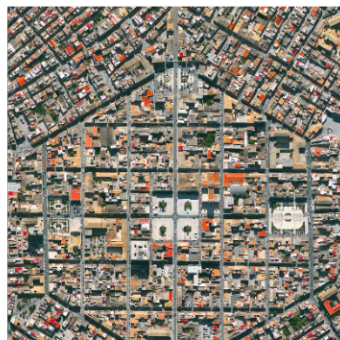
(a) Neural Axons



(b) Fungi Traces



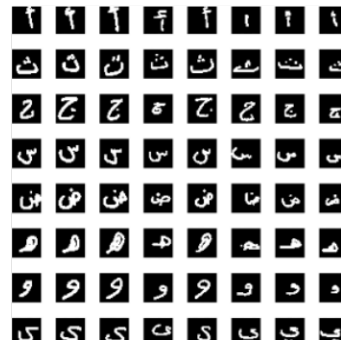
(c) Vessels



(d) Transportation Network



(d) Leaf Vein



(d) Handwritten Characters

Figure 3.1: Isotropic patterns occur widely in biology, medical, and general image scenarios, and relate to well-known challenges, such as vessel segmentation, geometric reconstruction, etc. Some patterns may not be fully isometric but do have equivalent semantic meaning for the targets at specific orientations. MASC unit intends to provide an efficient solution to these problems in a way that is rotational in-variant and scale in-variant. The model is fully trainable.

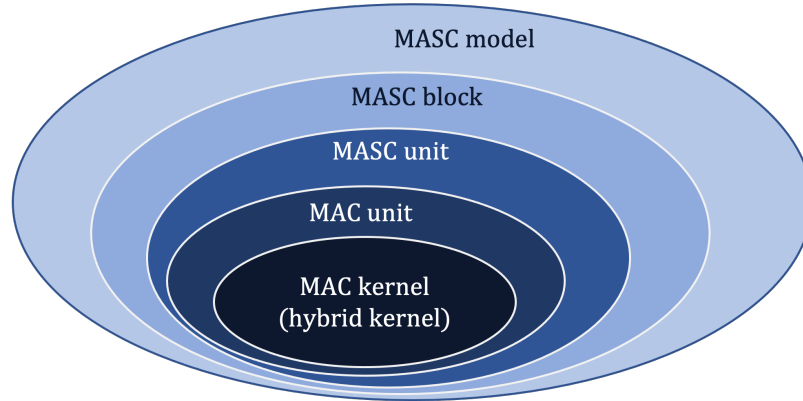


Figure 3.2: The concepts that will be introduced in this chapter as well as their relationships. A MAC kernel (hybrid kernel) is the most basic structure in this framework, and a MASC model is its ultimate form.

The methodology introduction in Section.3.2 generally follows this pyramid.

In this section, we test the performance of MASC model also on retinography datasets which share the same intrinsic pattern as axons – curvilinear patterns. More importantly, both tasks have nice orientation and width variation. Retina datasets are relatively easier to be obtained from the Internet, and public datasets such as CHASE-DB1 and DRIVE are commonly considered benchmarks for image segmentation. They are chosen as surrogate tasks in the experiment section of this chapter. These datasets help us to compare MASC model with other methods in terms of accuracy and efficiency.

## 3.2 Algorithm

### 3.2.1 Multi-Angle Convolution: Introduction

A MAC unit applies a set of kernels, each constructed to approximate a rotated version of basis, and selects the strongest response at each pixel. It thus picks out features at any orientation.

Suppose that  $\mathbf{w}_i$  is a vector of elements in the kernel  $i$ , which are normalised so that  $|\mathbf{w}_i| = 1$ . Suppose that  $\mathbf{x}$  is the vector corresponding to an input patch of the same size as the kernels, and  $\mathbf{x}_{3 \times 3}$  is a vector of the  $3 \times 3$  elements at the centre of the input patch.

Let matrix  $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_n)$  contain the elements of all  $n$  kernels.

Let  $\mathbf{m}_i = \mathbf{W}^T \mathbf{w}_i$ , a vector whose elements compare each kernel to  $\mathbf{w}_i$ .

The MAC output is calculated using the following steps:

- 1) Apply each kernel in turn to get an output vector  $\mathbf{v} = \frac{\mathbf{W}^T \mathbf{x}}{\|\mathbf{x}\|}$
- 2) Compare the shape of  $\mathbf{v}$  to that of each  $\mathbf{m}_i$  using  $R(\mathbf{v}) = \max_i \left( \frac{\mathbf{v} \cdot \mathbf{m}_i}{\|\mathbf{m}_i\|} \right)$
- 3) The output is given by

$$MAC(\mathbf{x}) = \|\mathbf{x}_{3 \times 3}\| R(\mathbf{v}) = \|\mathbf{x}_{3 \times 3}\| R\left(\frac{\mathbf{W}^T \mathbf{x}}{\|\mathbf{x}\|}\right) \quad (3.1)$$

Note that  $R(\mathbf{v})$  gives a strong response if the input patch is similar to one of the kernels,  $\mathbf{w}_i$ , in that the pattern of responses of  $\mathbf{x}$  to all the other kernels is similar to that of  $\mathbf{w}_i$ .

In the following we explain and justify the approach in detail, and describe how we can train a suitable set of kernels.

### 3.2.2 MAC: Hybrid Convolution

Borrowing the concept of matched filters [?, 6, 70], a convolution can be broken down into two parts. In Eq. 3.2, the kernel calculation is transformed to the product of signal strengths and the vector correlation. Here, the signal strength is computed by the corresponding sample norm. The kernel size not only influences the receptive field but also the output amplitude. The correlation term in the process is interpreted as a measurement of the similarity between kernel pattern and sample pattern. This interpretation is related to the fundamental motivation of MAC units of matching directional kernel pattern with oriented input pattern. To be more specific, as the rotation invariant module in MASC, MAC focuses on creating a group of normalised kernels that are trained to render strong response only to the inputs at particular orientations.

If  $x$  is the elements of the input patch (rasterised) and  $\mathbf{w}$  those of the weights of the kernel, then

$$C(\mathbf{x}, \mathbf{w}) = \mathbf{x} \cdot \mathbf{w} = \|\mathbf{x}\| \|\mathbf{w}\| \rho_{\mathbf{w}\mathbf{x}} \quad (3.2)$$

where  $\rho_{\mathbf{w}\mathbf{x}}$  is the normalised correlation between them.

Postulating both inputs and kernels as normalised before convolution, the computation will return 1 when the kernel pattern is exactly equal to the input, and the formula can be simplified to  $\mathbf{w} \cdot \mathbf{x} = \rho_{\mathbf{w}\mathbf{x}}$ . This reflects the principle of classic matched filters proposed for capturing radar pulses, where the focus is put on the shape of the signals. In our algorithm, the images of vessels and neural fibers are treated similarly by looking into the pattern shapes and response shapes. A matched filter can be constructed

only when having the knowledge of the shapes of all possible pulse series. The filter will then be convolved with the signals along the temporal axis, and a peak response is anticipated when target object reached.

In practice, target patterns are mixed with all kinds of deformation and noise. It is a challenging generalisation problem. In this case, an averaged filter is preferred, a model that deals with the noisy signal by compromising the algorithm sensitivity and the accuracy to some degree. Because of this moderate scores are rewarded to all kinds of target-like inputs. With gradient decent, the prior knowledge based crafting process of matched filters can be substituted by the iteration based approach from random seeds through the steepest path.

As mentioned in the introduction, random initialisation has raised people's concern from the aspect of model compression [21], and the anti-overfitting (as a consequence of large model) methodologies were studied such as dropout [108]. In semantic segmentation, the parameter volume for a decent model is usually above the general level due to its fine-grain classification nature, due to the demand of memorising more patterns. Even the vanilla U-Net is as large as 2.6M parameters [7], and its variants may use more [105].

Instead of treating oriented patterns like black boxes and relying on gradient descent to learn the pattern, a MAC unit is made to learn a principle pattern and deformation separately. We assume that there are  $n$  kinds of standard curvilinear patterns in  $k$  possible orientations.

Also inspired by Gabor CNNs [85], the hybrid kernel  $\mathbf{w}_\theta$  in MAC units is the Hadamard product (per-element) between a Gabor kernel  $\mathbf{g}_\theta$  and a normal convolutional kernel  $\mathbf{c}_\theta$  of the same size. The combined MAC kernel is defined as,

$$\mathbf{w}_\theta = \frac{\mathbf{g}_\theta \odot \mathbf{c}_\theta}{\|\mathbf{g}_\theta \odot \mathbf{c}_\theta\|} \quad (3.3)$$

where the Gabor elements  $\mathbf{g}_\theta$ , are defined as,

$$\mathbf{g}_\theta(x, y) = e^{-\frac{(x \cos\theta + y \sin\theta)^2 + y^2(-x \sin\theta + y \cos\theta)^2}{2\sigma^2}} \cos\left(2\pi \frac{x \cos\theta + y \sin\theta}{\lambda}\right) \quad (3.4)$$

The parameters for both parts can be updated through back-propagation. Examples of learned MAC kernels are visualised in Fig. 3.4. In the equation, kernel  $\mathbf{g}_\theta$  focuses on learning the ridge shape of curvilinear pattern at angle  $\theta$ . The second part  $\mathbf{c}_\theta$  learns the local deformation distribution and the other pattern variation of objects. The weights in  $\mathbf{c}_\theta$  can be updated freely.

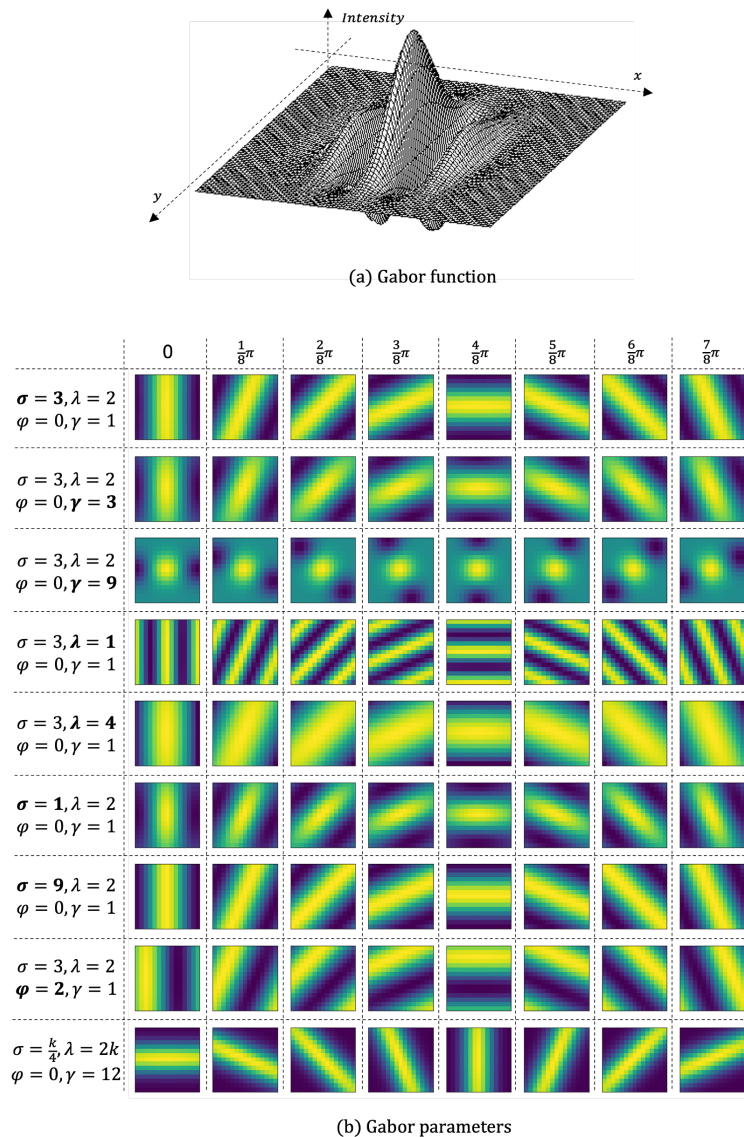


Figure 3.3: (a) A visualization of Gabor function which can be interpreted as the Gaussian expansion of a sinusoidal wavelet function on a 2D plane. (b) Gabor function is controlled by four parameters, including  $\sigma$ (variance),  $\lambda$ (wavelength),  $\phi$ (phase offset), and  $\gamma$ (x/y ratio). Updating the parameters can adjust the patterns to better fit to the targets. The last row demonstrates a typical initialization of the Gabor elements in MAC hybrid convolution, in which  $k$  means the kernel size.

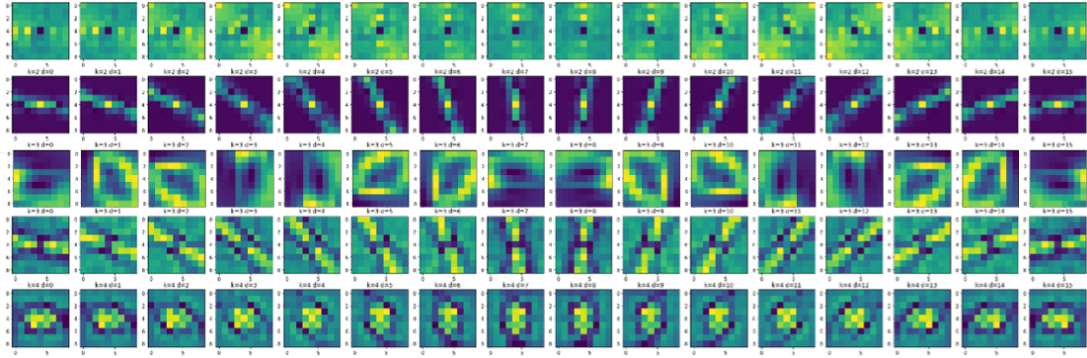


Figure 3.4: Some visualised examples of hybrid kernel patterns. These kernels are acquired from 16 directional  $9 \times 9$  MAC units. Though initialised as rotational ridges (similar to Gabor filters), the patterns after training are found not necessary following the restriction of  $\mathbf{G}_\theta$  but still present in a rotational relationship. The free-updatable element  $\mathbf{C}$  plays a critical role in this change.

In the function, parameter  $\gamma$  controls the variance ratio on the two perpendicular axes,  $\sigma$  represents the variance of the Gaussian controlling the intensity and  $\lambda$  defines the cycle period. Specifically, parameters  $\sigma$  and  $\lambda$  jointly describe the intensities spreading along the ridge direction at angle  $\theta$ . Some example patterns generated with different parameter values are displayed in Fig. 3.3 (b).

The directional convolution generates the feature ready for comparison. But due to the rectangular window shape, the overall intensity distribution of each  $\mathbf{w}_\theta$  is different. For example, for ridge patten, the diagonal oriented kernel's norm is significantly greater than the vertical/horizontal oriented ones. Thus  $\mathbf{W}_\theta$  is normalised to ensure a fair competition between the directional responses.

### 3.2.2.1 Why choose Gabor function as constrained basis?

There are four reasons for the choice:

1. The model is expected to cover all the directions - the Gabor function can be tuned for any angle.
2. The function is relatively simple and thus parameter efficient four core parameters can control the kernel weights at any size.
3. A ridged shape can be constructed which matches the object shape and constrains the variation thus encourages a stabler evolution for the hybrid kernel.

4. Gabor function is differentiable, so that the parameters can be tuned through training.

This hybrid kernel strategy makes sense with curvilinear objects, where modelling a targets' basic form and variations separately requires fewer parameters. For a more general dataset more MAC units with larger kernel sizes may be necessary.

We compare an input patch,  $\mathbf{x}$ , with each kernel,  $\mathbf{w}_i$ , in turn using

$$\mathbf{v}_i = \frac{\mathbf{w}_i \cdot \mathbf{x}}{\|\mathbf{x}\|}, \forall i \quad (3.5)$$

Here,  $\mathbf{v}_i$  is the directional response at direction  $i$ . We probe an input pattern with all  $n$   $\mathbf{w}_i$  and store the results, giving the directional response vector  $\mathbf{v}$ . We postulate that each  $\mathbf{w}_i$  normalised, so  $\|\mathbf{w}_i\| = 1$ . Since  $\|\mathbf{x}\|$  is in the denominator, amplitude differences are eliminated from the entire process. Therefore the elements in  $\mathbf{v}$  are normalised correlations. Though this hybrid convolution puts the focus on the patterns, the semantic amplitude information wrapped in  $\|\mathbf{x}\|$  is not abandoned and will be integrated into feature later (see Section 3.2.7).

As a result of this step, a MAC kernel has a clear output intensity range of  $\forall \mathbf{v}_i \in [-1, 1]$ . In a standard MAC, no additional batch/layer normalisation is needed after the hybrid convolution. The MAC process shares some similarities with self-attention unit (SA) [30].

It is important to keep the kernels rotated consistently. A problem is that training samples may be very imbalanced from the aspect of orientation. It is very likely that the vessel in an organ is influenced by gravity and occurs mostly in certain orientations. The condition also happens on axons, which are guided by the bio-medical cues as a grid. The frequently occurring directions can dominate in training, so the rotational relationship is broken. Potentially data augmentation can help this by increasing the quantity of rare samples. Or, if we have sufficient awareness of the orientation distribution, we can give more weight to the gradients on the rare orientations. However, such approaches are mitigations which do not solve the problem fundamentally. Here we propose a different method.

A portion of information is shared between the directional kernels, which can be used to further condense the parameters. Inspired by the Steerable Convolutional Network [25], we construct kernel groups and make kernels internally linked. This construction method is described as filter reuse. We only need one quarter of the number of parameters and it boosts the robustness of the model.



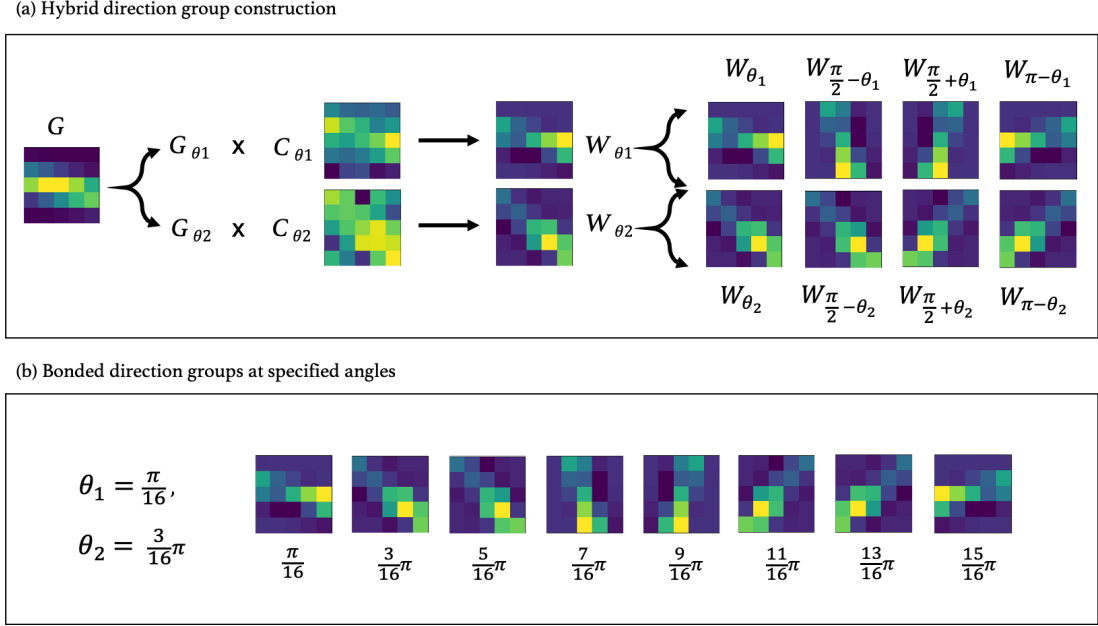


Figure 3.5: The process of constructing a hybrid kernel set of 8 direction in MASC. It has two elements, where  $\mathbf{G}_\theta$  is generated by a trainable Gabor function and  $\mathbf{C}_\theta$  is free-updatable. Hybrid directional kernels are managed under a re-use scheme using matrix transpose and  $90^\circ$  rotations as inspired by [25]. The kernels are bonded by the re-use rules and share weights and gradients. By assigning base kernels to precise angles, a set of evenly distributed directional kernels can be maintained.

### 3.2.3 MAC: Kernel Reuse

The kernels at specific directions are designed to share weights in the MAC, in a way similar to that used in steerable CNNs [25] and Group CNNs [70] [20]. For example, suppose that there are two directional kernels of  $\theta_1$  and  $\theta_2$ . Through filter transformations (transpose,  $90^\circ$  matrix rotation), these two base kernels can be extended to two groups of kernels of searching the range  $[0, \pi)$  radians,  $\{\theta_1, \frac{\pi}{2} - \theta_1, \frac{\pi}{2} + \theta_1, \pi - \theta_1\}$ , and  $\{\theta_2, \frac{\pi}{2} - \theta_2, \frac{\pi}{2} + \theta_2, \pi - \theta_2\}$ . With another round of matrix rotation, the ranges can be extended to  $2\pi$ .

If we can find a method precisely lock the two base kernels at  $\theta_1 = \pi/16$ ,  $\theta_2 = 3\pi/16$ , then simply by combining the two groups, a  $\pi$  (or  $2\pi$ ) space can be evenly split into 8 (or 16) angles. As the kernel weights are internally linked, the gradient flow through one angle will also be applied to the evolution of the other three extensions. Simply by adding more base kernels, the set of angles can be constructed to reach arbitrarily small intervals.

In later experiments we show that increasing the number of orientations beyond 8

leads to diminishing benefits.

This is probably because a finer splitting of angles means great information repetition between adjoining kernels. It is also the case that a convolutional kernel is a discrete function. In MAC, it is interpreted as an emulation of target relative patterns, where small directional nuance may not be faithfully expressed under the grid nature.

The response shaping (in Section 3.2.4) ensures that the intra-group relationship is stable.

The choice of transformations of each basic kernel is task dependent. Other distortion operators, e.g. translation, are not useful in this case. On the one hand, a transformation such as translation is equivalent to the parameter  $\phi$  in MAC Gabor function. On the other hand, the local pattern of vessel objects are ridged shaped and symmetric and sampled by a moving window.

For an object having symmetric character (about origin/central line), transpose and rotation are appropriate choices. The number of rotation folds is decided according to the image nature.

### 3.2.3.1 Justification for the Kernel Re-use Scheme

Assuming perfect symmetry, using four  $90^\circ$  rotations will repeat the pattern, as  $\text{rot}90(\mathbf{x})/\text{rot}180(\mathbf{x})$  will be the same as  $\mathbf{x}/\text{rot}270(\mathbf{x})$ . For the other cases, such as a non-asymmetric but isotropic pattern (cells, bones, etc.), four-fold rotation and transpose can be appropriate. A similar horizontal flip operation is more popular [25] [70].

In a MASC designed for axon and retina vessel objects, four-fold  $90^\circ$  rotation and transpose operations are used to produce a 16 direction-set. The rotation space is limited within  $2\pi$  instead of  $\pi$ . The predicted pattern repetition problem is also confirmed by an experiment. The charts in Fig. 3.6 demonstrate the internal correlation distributions. The right column with  $\pi$  range (8 angles, interval= $\pi/8$ ) has the correlations presented as a single peak wavelet. The directional kernels after training are more similar to the closer ones but different from the kernels having large angle difference, which is consistent with the theoretical presumption. We do observed that the situation of  $2\pi$  (16 angles, interval= $\pi/8$ , left column) does not work in the same way. Each row of it could have two peaks, and the phase distance between the peaks is exactly  $\pi$ . However, the two-fold rotation allows kernel to learn small tube breadth changes, as the retina vessels are not completely symmetric about the center. In a further study about this choice, we found , for the  $2\pi$  scheme, pattern repetition is some degree persistent during training. But the patterns are discriminable during response shaping

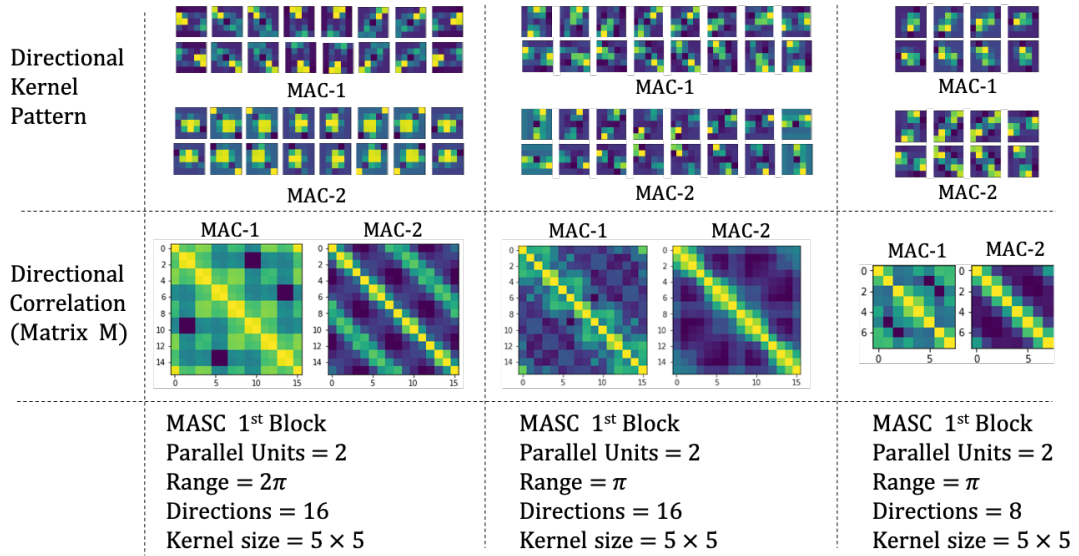


Figure 3.6: From left to right: 1) 16 angles in range  $[0, 2\pi)$ , 2) 16 angles in range  $[0, \pi)$ , 3) 8 angles in range  $[0, \pi)$ . Each visualised correlation between directional kernels (entries in key matrix  $\mathbf{M}$ ) is calculated by  $\mathbf{w}_i \cdot \mathbf{w}_j, \forall j$ . The first and third cases spread the kernels in the directional space with  $\frac{\pi}{8}$  intervals. The second case (middle) has an interval of  $\frac{\pi}{16}$ . The colours in each  $\mathbf{M}$  matrix plot indicate how strongly the directions are correlated. As curvilinear targets are not perfectly center-symmetric, using a  $2\pi$  range will lead to multiple correlation peaks in one row but not necessary cause confusion in directional selection, as shown by the examples in the first column. A MASC unit can still learn to find the most matched pattern at the right orientation even there is a close competitor initially.

and the optimal selection is functioning. Correct gradients are produced to update  $\mathbf{C}_\theta$ , covering a doubled number of directions with no additional parameter.

For the case having  $4a$  orientations, the base filters are set to the directions of,

$$\theta_j = \frac{(1 + aj)\pi}{8a}, j \in [0, a), j \in Z \quad (3.6)$$

Filter re-use enables a significant reduction to the parameter numbers.

As claimed, to facilitate a set of rotational filters with arbitrary angles, several groups of intrinsically correlated directional filters are put together. Using more base in kernel re-use can make the model do a finer survey. Though, in an ideal situation, a perfectly direction-balanced training set is just enough to keep a sequential rotation relationship. In practice, MASC models typically need more to encourage rotation. Their strength comes from two sources, response shaping and the rotation secure term in the cost function.

### 3.2.4 MAC: Response Shaping

Response shaping is a method designed to ensuring that the output of the MAC is quasi-invariant to rotation. It is applied after the directional responses are generated.

The simplest way of achieving near-invariance to rotation is by taking the maximum response from a set of kernels which represent rotated versions of some basis. Similar processes can be found in many other equivalent pattern scanning works [36] [135] [81]. However, this method is not very ideal as it has three intrinsic drawbacks, which has also been suggested by the experiment results displayed in Table.3.10.

First, the gradient under this method only flows through the selected kernel, which means only one of the kernels is updated by each example. This process is inefficient as only a small portion of weights are impacted, and leads to the risk of exaggerating the semantic gap among the directional kernels. This is not appropriate for a learning based model where balanced training is important. Second, the max function here requires the selected direction to be the most significant but does not specify how significant. A maximum response of 0.5 has a different meaning to a maximum of 1. Besides these, the responses from directions are also expected to follow an anticipated order. The responses from orthogonal angles should not be close in value given a curvilinear pattern.

Borrowing a concept from graph models, the problem of matched direction selection can be transferred to a message integration step gathering information from the

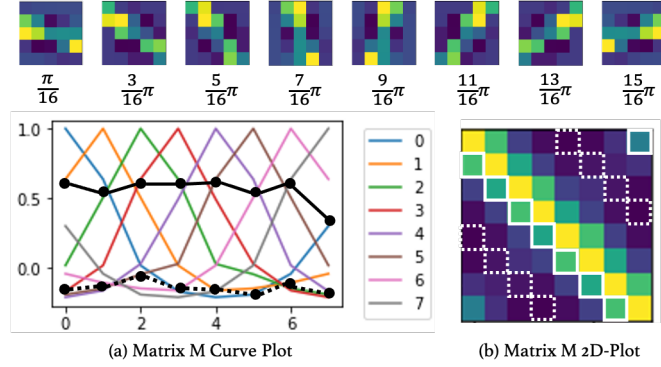


Figure 3.7: The first row illustrates a set of directional kernels after training. Chart (a) shows the estimated 'pseudo-ideal' response shapes from their  $\mathbf{M}$  matrix. Different colours in (a) distinguish the shape curves from different angles. The elements with phase offset  $\frac{\pi}{8}/\frac{\pi}{2}$  are connected by the black dash/solid lines. Plot (b) visualises the chart (a) in 2D, in which rows and columns represent the angles, and the black solid/dash angles are turned in the white black solid/dash line enclosed squares. A weight vector  $\mathbf{w}_\theta$  is found closely correlated with the kernel with a small phase offset, and almost perpendicular to the one at the orthogonal angle.

connected directional nodes. We are expecting the function to meet following requirements, (i) it takes the message from all neighbours and (ii) has an explicit response expectation. Based on these two factors, we developed the primary form of response shaping.

The principle of response shaping is combining the responses each angle  $\mathbf{W}_\theta$  and comparing their shape (output vs angle index) with the ideal shape. In practice, the ideal directional responses are unknown thus must be estimated.

As with matched filters [70], an ideal detector should be very close to the target. Thereby if assuming a target pattern at orientation  $i$  was approximated by a hybrid kernel  $\mathbf{w}_i$ , the response of this pattern with detector  $\mathbf{w}_j$  can be estimated as  $\mathbf{w}_i \cdot \mathbf{w}_j$ . Repeating this process on all directions, the pseudo-optimal responses matrix  $\mathbf{M}$  is estimated by the covariance matrix of weight vectors, which explicitly suggests the expected value from each direction, defined as

$$\mathbf{M}_{ij} = \mathbf{w}_i \cdot \mathbf{w}_j, \forall i \forall j \quad (3.7)$$

Symbols  $\mathbf{w}_i$  and  $\mathbf{w}_j$  can be any two hybrid kernels in  $\mathbf{W}_\theta$ .

Some visualised examples of  $\mathbf{M}$  can be found in Fig. 3.7. Typically, the matrix will have ones on the diagonal and higher values near the diagonal. This is because the two consecutive directional kernels are anticipated to have similar patterns.

The  $\mathbf{M}$  matrix generated by response shaping can provide some interpretability about the status of MASC model. For example, the rotational effect can be read from the standard deviations of all  $\mathbf{M}_{i,i+k}$  entries. It helps understand and manage the training procedure, as the function of a MASC unit and its converged shape is well-defined and anticipatable.

The preferred comparison method or message integration function here is dot product, where the more similar two distributions are the higher the score will be. It encourages the kernel to shape its directional response towards the estimated shape.

We also test several other functions as mentioned in [53], and found the current dot-product approach gives the best and most robust performance. Other functions and results of evaluation can be found in the experiment section.

The process of shaping is matching the directional response  $\mathbf{v}$  with every row in  $\mathbf{M}$ . The maximum operation is used to determine the matched direction. Different from simple maximum, the response shaping compares the distribution of response with  $\mathbf{M}$ , thus every channel has gradient flow through during back-propagation and all kernels will be updated. The output of the shaping is given as

$$R(\mathbf{v}) = \max_i \left( \frac{\mathbf{v} \cdot \mathbf{m}_i}{\|\mathbf{m}_i\|} \right) \text{ where } \mathbf{v} = (v_1 | \dots | v_n)^T, \mathbf{v}_i \in [-1, 1] \quad (3.8)$$

In the case of perfect rotation, a row in  $\mathbf{M}$  should be identical to all the other rows only with position shift. However, in practice, there could be lagging during the pseudo-rotation evolution, and also correlation inconsistency could exist from one direction to another due to the square shape of kernel. It is necessary to divide the dot product by  $\|\mathbf{m}_i\|$  to account for the amplitude difference.

#### 3.2.4.1 Why not normalise $v$ in response shaping?

It is observed that  $\|\mathbf{v}\|$  varies from input to input, and this may cause  $R(\mathbf{v})$  not to reflect the shape of responses. This suggests that the response shaping process ought to be divided by the norm of  $\mathbf{v}$  as well. As shown in Eq.3.27, if given both  $\mathbf{v}$  and  $\mathbf{m}$  normalised, response shaping can be simplified to a squared difference equivalence. However, after a cross-validation test on the both cases (dividing and not dividing by  $\|\mathbf{v}\|$ ), a significant difference in the results is found with the un-normalised case being better.

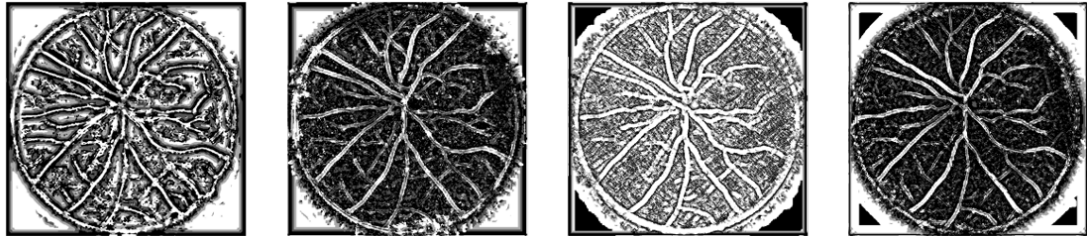
There appear to be three reasons for this:

1. As both  $\mathbf{w}$  and  $\mathbf{x}$  has been normalised before calculating  $\mathbf{v}$ , the variance of  $\|\mathbf{v}\|$

(a) Raw image and ground truth



(b) Pyramid representation maximum index map



(c) Response shape maximum index map

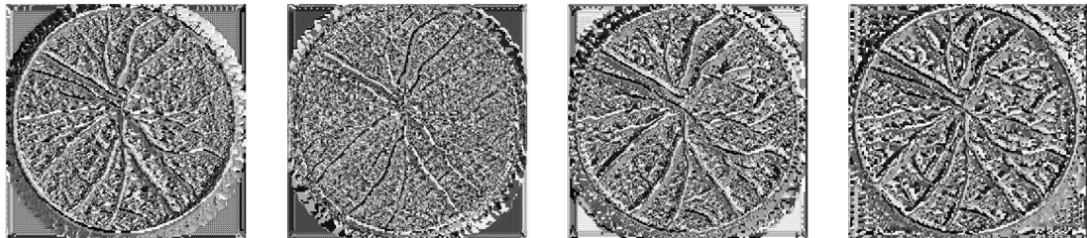


Figure 3.8: Some visualised examples of maximum index maps with the raw image given on the left. There are two rounds of channel selections in a MASC unit, first for orientation matching, second for scale matching. In the scale index maps, lighter colours represent patterns searched on a larger scale. In the orientation index maps, different colours represent different directional kernels used. The orientation indexes change smoothly with the pattern direction. The same effect can also be observed from the rim of the eyeball. The scale indexes have a certain degree of consistency which follows the thickness change of the retina vessels. These index maps contain valuable information which can be fed into the model after appropriate transformation.

is small.

2. Non-target inputs can beat the target pattern in response shaping only when the response are averagely high. However, in practice, this unlikely. A input pattern cannot be strongly correlated with all directional kernels, given the directional kernels themselves are sparsely distributed in the weight space.
3. The function with  $\mathbf{v}$  unnormalised encourages the cases with direction-concentrated strong responses.

Thus the probability of having  $\|\mathbf{v}_{target}\| > \|\mathbf{v}_{non-target}\|$  is high. Normalising  $\mathbf{v}$  in the equation is actually decreasing the response scores of target patterns and boosting the non-target pattern score overall.

The index of the maximum signal on depth direction indicates the best matched rotational kernel of the local pattern. Some examples of the index output can be found in Fig. 3.9. The target regions are usually presented as constant choices, when irrelevant areas will have more random indexes. Also, the bending parts usually have a very smooth transition as zigzag conditions are rarely observed in these tasks. When a target trace changes its orientation, the maximum shape index also changes consistently. These index maps contain useful but discrete information which can be featurised by folding.

In the literature of attention in language processing, in order to make the positions with stronger attention scores stand out, exponential was used to exaggerate the high signals. A similar strategy was also found helpful in response shaping. Here we use a revised function:

$$\mathbf{v}' = \exp(c\mathbf{v}), \text{ where } \mathbf{v} = (v_1 | \dots | v_n)^T \quad (3.9)$$

$$\mathbf{m}'_i = \exp(c\mathbf{m}_i) \quad (3.10)$$

$$R'(\mathbf{v}) = \max_i \left( \frac{\mathbf{v}' \cdot \mathbf{m}'_i}{\|\mathbf{m}'_i\|} \right) \quad (3.11)$$

This brings some variation to the shape comparison and shifts more attention to the matched direction. The constant parameter  $c$  controls the degree of focusing. As



$c$  increases the function converges to a simple maximum. Compared to the original formula, this modification is found to lead to faster convergence.

### 3.2.5 MAC: Update $\mathbf{M}$ with Momentum

During training the estimated response matrix  $M$  is updated in each step, which makes the shaping plan dynamic. Compared with the straightforward updating ( $a = 0$ ), momentum is often used to stabilise the training process, as,

$$\mathbf{M}_t = a\mathbf{M}_{t-1} + (1 - a)\mathbf{w} \cdot \mathbf{w}^T \quad (3.12)$$

The  $\mathbf{M}_t$  is the momentum adjusted matrix at step  $t$ . It constitutes two factors,  $\mathbf{w} \cdot \mathbf{w}^T$  and  $\mathbf{M}_{t-1}$ , And  $a \in (0, 1]$  is the momentum coefficient. The record term  $\mathbf{M}_{t-1}$  is from the previous step, therefore gradient only flows through  $\mathbf{w}$ . This momentum strategy enables a smooth evolution of  $\mathbf{M}$ , which is particularly important at the early stage of training, during which parameters can change quickly.

We varied the value of  $a$  in the experiments, and found a rather large value, such as 0.99, can provide a stabler transition, similar to that suggested in MOCO [46] for constructing contrast learning models. Assigning a very small weight to  $\mathbf{w} \cdot \mathbf{w}^T$  makes the gradient on  $\mathbf{M}$  have less impact than that on  $\mathbf{v}$  during the back-propagation.

An individual MAC unit is designed to deal with patterns at different orientations, but assumes a fixed scale. We now extend the approach to deal with scale variation.

### 3.2.6 MASC: Pyramid Representation

The MAC algorithm can distinguish target in different orientations but has a postulate of fixed thicknesses. Comparing to the scale of target patterns, the typical convolution window size is smaller. A vessel can have a width from 3 pixels to 12 pixels. Henceforth, enlarging the window size to match object is not the first choice since the cost will increase exponentially.

It is common in segmentation tasks that a model uses a pooling layer to get a larger receptive field, e.g. U-Nets and [106]. Inputs are down-sampled, and every detector works on the results of previous detectors. The pattern selected at each level is not same. Recently, more and more works shift their focus to pyramid pooling. Similarly, pyramid pooling [38] [1] employs a set of pooling operators to down-sample feature

Pyramid Pooling in MASC

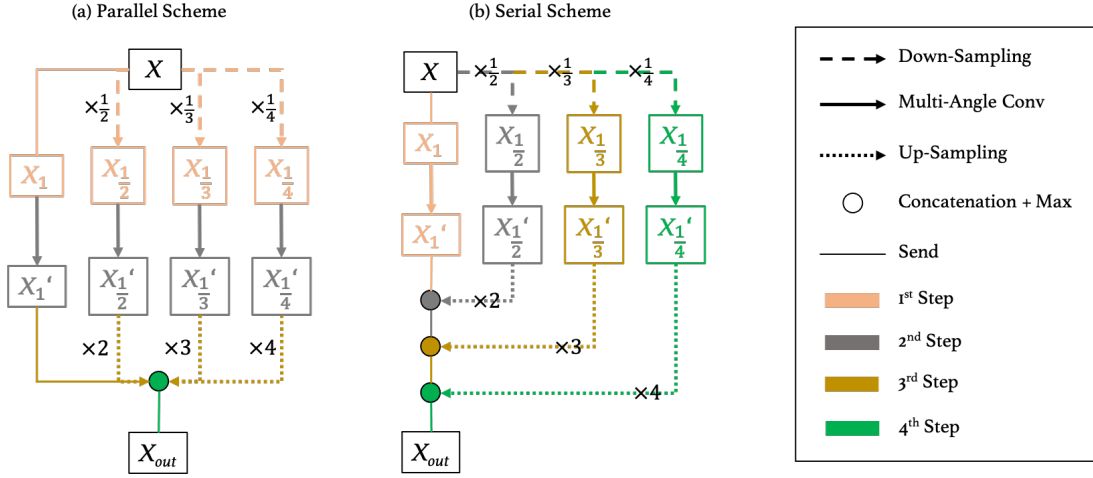


Figure 3.9: A demonstration of two computation approaches, pyramid pooling in parallel and in serial. The colours indicate the execution order, and the lines with different dash and arrow styles represent different functions. The parallel scheme is time-efficient with GPU, and the serial scheme is memory-efficient and usually takes longer.

maps. The scale operators are not arranged sequentially but in parallel. However, most of its applications are aiming at independent search on each lane.

In MASC, we utilise these representations to address the scale variation of target pattern. The same set of directional detectors will be applied to each representation. After the orientation selection, the result maps will be projected back to the original size via bilinear interpolation. Pattern intensities are aligned at grid positions, and the maximum channel along the scale direction is selected. Two important pieces of information are acquired, a scale-invariant feature map, and an index map depicting the thickness change of target pattern. Illumination conditions and shadows may cause discontinuities in pattern texture. Down-sampling features can mitigate this issue by overcoming image flaws [134] [39].

The pyramid ratios are hyper-parameters and usually set to be  $\{1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}\}$ . This requires the input image width and height to be  $12 \times k, k \in Q$ . It is achieved by adding 'reflect' padding before processing.

### 3.2.7 MASC: Re-Scale Feature Maps

The signal strength, estimated by norm, is removed in the step of hybrid convolution so that competition among scale channels is fair. These values are not going to be

		Directional Response	
		High	Low
Intensity Strength	High	Very high	High
	Low	Low	Low

(a) Common Re-scale

		Directional Response	
		High	Low
Intensity Strength	High	High	Low
	Low	Low	Low

(b) Alternative Re-scale

Figure 3.10: The case studies of two re-scale schemes to give a general comparison. The terms 'high' and 'low' in directional response maps typically mean values (rebalanced in the alternative scheme) above or below 0. The plot shows the current re-scale scheme can better separate the classes. There is not a constant standard for strength maps as it varies from case to case.

abandoned. There are normal convolutional layers in a MASC model providing other semantic analyses. This semantic information is expressed by the signal strength carried by  $\|\mathbf{x}\|$  in Eq. 3.5. After the pattern orientation analysis, this information is fed back to the feature flow via a re-scaling procedure just after the maximum scale channel selection.

$$MAC(\mathbf{x}) = \|\mathbf{x}_{3 \times 3}\| R'(\mathbf{x}) \quad (3.13)$$

Here  $\mathbf{x}_{3 \times 3}$  is a vector containing the  $3 \times 3$  elements at the centre of the target patch.

Instead of collecting values from scaled feature maps, a uniform signal strength estimator will be employed to acquire the norm values from the input features of a MAC unit. Fine-grain contrast is expected to be necessary, so we use a  $3 \times 3$  window. The collected intensity strength map is then multiplied with the directional response map to generate the output of a MASC unit.

### 3.2.7.1 Alternative Re-Scale Practice

There is an implicit problem in the introduced common re-scale formula. The distribution of the sampled intensity strengths and directional response values are different, causing potential divergent semantic meanings behind two equivalent MASC outputs. The range of directional responses is between  $[-1, 1]$ , and the intensity strengths are

in  $(0, +\infty)$ . Both the distributions are skewed and biased but in very different degrees. The strength map typically has a heavy long tail, though the median is relatively low, when the directional response map is just moderately skewed to the right with a positive median (typically  $0.1 \sim 0.3$ ). This causes a situation when using the original multiplication scheme, an obtrusive area with irrelevant pattern could still be strong after MASC, which should be filtered. To adjust this difference and produce a more constant and representative MASC result, we proposed an alternative rescale scheme, written as Eq. 3.15. Here  $\sigma(\cdot)$  is a sigmoid activation function.

$$S(\mathbf{x}) = 2\sigma(a\|\mathbf{x}_{3\times 3}\| + b_1) \quad (3.14)$$

$$MAC(\mathbf{x}) = S(\mathbf{x})^{R(\mathbf{x})+b_2} = e^{\log(S(\mathbf{x}))(R(\mathbf{x})+b_2)} \quad (3.15)$$

The distribution means of the strength maps is firstly adjusted by a bias term, then project to  $(0, 2)$  using a double ranged sigmoid function, after which the long tail part is compressed. With the power of adjusted  $R(\mathbf{x})$ , the function ensures only the case of both high intensity strength and high pattern correlation can produce a high MASC score. Norm strength estimator is still used here, and the input  $\mathbf{x}$  is pre-rectified by ReLU. The learnable bias term  $b_1$  and  $b_2$  provide some control over the distribution. Together with  $\sigma(\cdot)$ , shift the disparity point to a proper level. The negatively intensive areas are eliminated, making desired patterns distinguishable from irrelevant ones.

With this alternative re-scale function, the problems such extreme values are mitigated, shown in Fig. 3.11. Case (a) illustrates the effect on some imbalanced input features. Extreme strong intensities are given to some areas by the prior layers, and these strong spots are rebalanced in the result maps which generally follow the directional response scores. Case (b) has noisy input features, and this is succeeded by the semantic strength map. The result shows the function is able to gather and combine the merits of both maps. In some situations like Case (c), the prior layers fail to detect some subtle structures. In the meantime, the pattern-based MAC unit is very sensitive to subtle textures thus able to extract them though with noises. With the adjust bias in the function, a MASC model can retrieve these weak signals. A similar noise-control outcome can also be seen in Case (d) where the function balance the noise in the MAC response map with the evidence from the semantic strength map. The case comparison between the two practices can be found in Fig. 3.11.

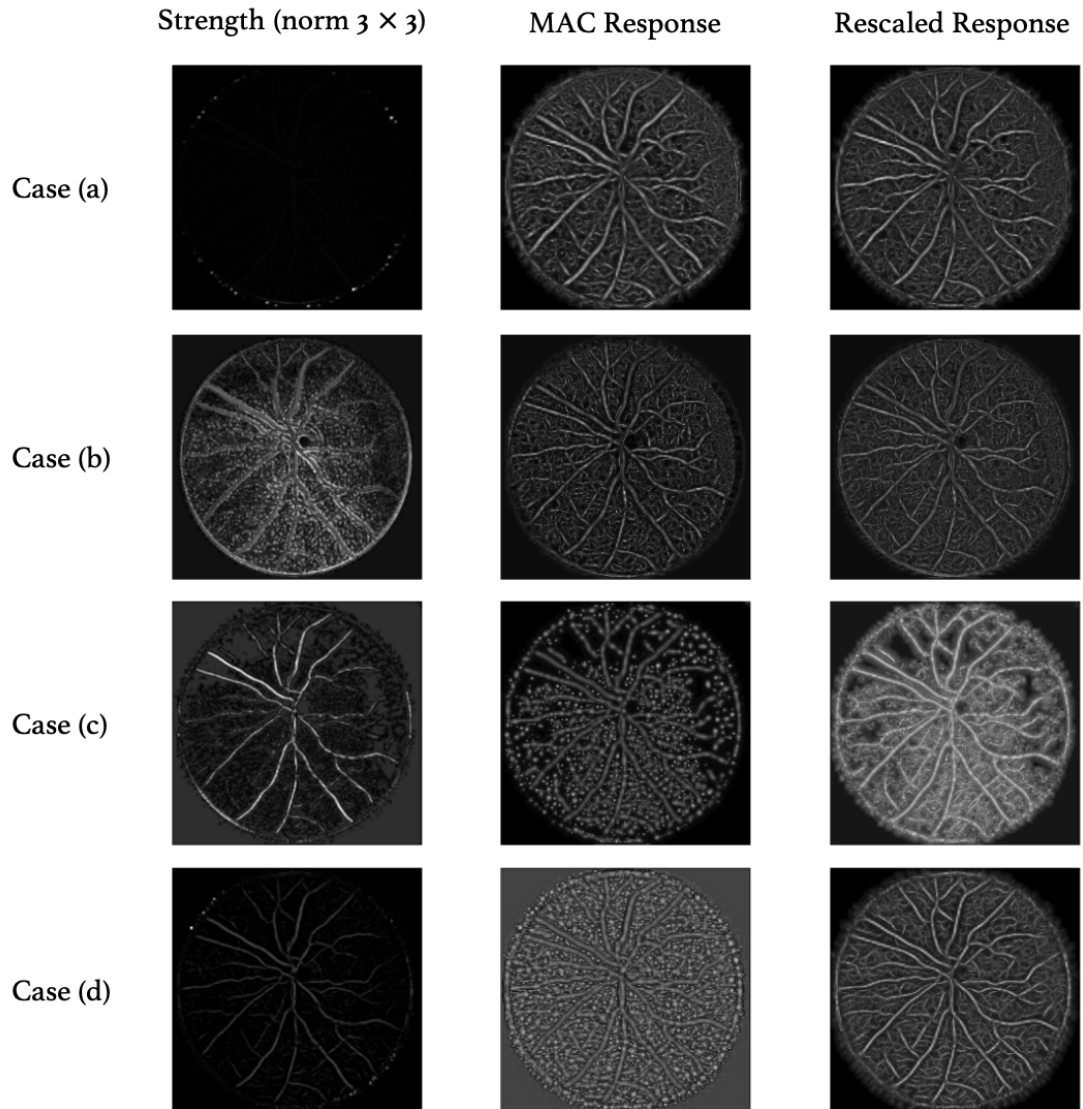


Figure 3.11: The MASC outputs using the alternative re-scale scheme. The rows above represent each individual case, and columns are the input norms (feature strength maps), the outputs of MAC(directional responses), and the results after integration. Four examples are showing above, (a) an imbalanced feature strength map; (b) a noisy feature strength map; (c) a detail-lost feature strength map; (d) a noisy directional response map.

The whole structure is called Multi-Angle and Scale Convolutional (MASC) Unit. Unlike the other steerable kernel methods [25] [125] based on homogeneous kernels or rigid rotation mechanism, training-based hybrid kernels with pyramid representations strategy are used instead to achieve a rotation and scale in-variant effect. It is found very economical in parameter usage, and subtle image flaws (negative factors to thin structure segmentation) are mitigated, as obstacles and small breaks are eliminated in low resolution representation.

The hybrid kernels are applied on each pyramid representation of the input. As shown in Fig. 3.12(c), the outputs are then upsampled back to the input scale via bilinear interpolation. Unlike using distribution to maintain kernels' rotational relation, the uneven updating is not a problem for addressing scale variation in which only one set of kernels are used. The features from different scales are summarised by a maximum function at the end. The index of the maximum signal on depth direction indicates the thickness of the local pattern, this is confirmed by the evidence in Fig. 3.10. Just like the response shape index map, the indexes over target area have a more constant pattern and a smoother transition than the background area.

### 3.2.8 Building Block Construction

The flowcharts in Fig. 3.12(a) shows the architecture of a MASC model. The model is built by stacking a series of MASC Blocks. Illustrated in Fig. 3.12(b), the input of a MASC Block will be firstly compressed from 8 channels to 2 channels. These feature maps will then be sent to a normal convolutional trajectory and a MASC trajectory. This structure is influenced by the initial idea of MASC of a curvilinear attention gate. Combining the normal convolutional and residual features with MASC maps, the outputs will then be mixed and projected back to 8 channels and ready to be sent to the next level.

The intuition behind the block design is finding a way of addressing the input constraints of a MASC unit which has to be single-channel and saving parameters at the same time. Supported by the works and experiments of the invert bottleneck which was first proposed in MobileNets [52] [103], the parameter efficiency is pushed further in our algorithm. A MASC block is designed to be wide-thin-wide and has independent feature groups.

Just like multi-head attention models [113] [140], in a MASC block, multiple MASC units are arranged in a parallel manner learning from different types of isotropic patterns. There is a pre-MASC layer generating  $n$ -channel feature maps, here  $n$  denotes

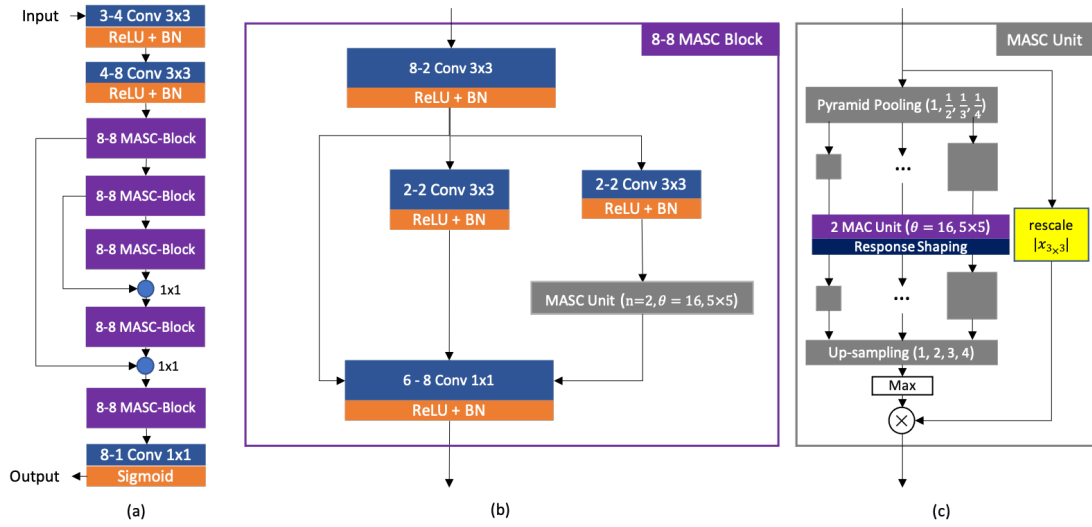


Figure 3.12: (a) Model MASC-5-2-8, with 5 stacking MASC Blocks, each with 2 parallel MAC Units of 8  $5 \times 5$  directional kernels; (b) 8-8 MASC Block, with 8 input and 8 output channels and a thin bottleneck; (c) MASC Unit, including 2 MAC Units applied on pyramid representations, where only the maximum is taken from across the scale channels. MASC is a normalized calculation, the result value will be rescaled with a  $3 \times 3$  L2pooling node. This structure is built for axon and retina segmentation tasks, modification could be necessary of the other tasks.

the number of the parallel MASC units. This can provide some degree of feature pre-processing for each pattern detector.

We name the model having 5 MASC blocks, each has 2 independent 8-directional MAC units, as a MASC-5-2-8 model. An example of MASC-5-2-8 can be found in Fig. 3.12. A single MASC unit with this configuration can work like 32 filters (on 8 directions and 4 scales).

### 3.3 Model Structure

A MASC embedded model is constructed using MASC building blocks. And each MASC blocking carries one MASC unit. Every MASC unit has two elements, a pyramid representation module, and several groups of oriented detectors as known as the Multi Angle Convolutional units (MAC units). Two challenges, scale and orientation invariance, are dealt with by these two parts. The general structure and data flow are illustrated in Fig. 3.12.

Though the model is trained in an end-to-end fashion, a MASC model has good interpretability. It is designed in a modular way, where each module is developed for

a clear purposes. The definition and relationship between the intermediate metrics and the ultimate output are visualised in the corresponding section, as well as the method of interpreting them. A MASC unit is not only for solving a segmentation task. More importantly, it provides a methodology for generating and manipulating orientated kernels in a network, which is still an area of active research in the field.

## 3.4 Experiments

In this section, we test a MASC unit embedded model with two retina datasets, CHASE-DB1 and DRIVE, and the microscopy axon dataset made by ourselves.

### 3.4.1 Datasets

The retina vessel dataset CHASE-DB1 [96] contains 28 8-bit color eye fundus images of left and right eyes with a resolution of  $999 \times 960$ , sampled from 14 school children. The dataset is public and open for download. The package includes two sets of ground truths, where the annotations from the first observer(1stHO) are used for training and testing. The second set usually acts as a 'human' baseline. The images are separated into two groups. The training set has 20 images and the test set has 8 images. 8,000  $48 \times 48$  patches are randomly cropped from the training images. There is no standard FOV provided in the official release, so we created the masks by taking the non-zero areas in the raw images.

Another popular eye fundus dataset we used is DRIVE [109], created for diabetes diagnosis and screening, sampled from 25-90 years of age. The dataset has 40 8-bit color eye fundus images, which have been pre-split into two sets evenly for training and testing. Each image is in  $565 \times 584$  resolution, has the ground truth from two observers. The same 8,000  $48 \times 48$  patch-sampling scheme is used. A standard FOV is included in the dataset.

For the axon dataset, the original  $2200 \times 2200$  resolution microscopy video frames were downsampled to  $550 \times 550$ . We arbitrarily chose and annotated 30 images, 20 for training and 10 for testing. The training images follow the same procedures, randomly cropped into 8,000  $48 \times 48$  patches. Because they are grey-level images, different from the one in Fig. 3.12, the first layer of an axon model has 1 input channel instead of 3.

The default patch size may not be optimal for all tasks, we choose it as a trade-off



to experiment efficiency. Associated with the object scales, a better choice could exist. A training-validation ratio of 4:1 is used. The test set is kept separately.

### 3.4.2 Configuration

If without any specific statement, a default architecture of MASC-5-2-16 is used, its structure has been illustrated in Fig. 3.12. A MASC-5-2-16 is constructed by stacking 5 MASC blocks, each having 2 16-directional parallel MAC units applied on 4 scales, whose kernel size is  $5 \times 5$ . Its pyramid scale scheme is  $(1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4})$ . The default momentum coefficient of updating key matrix  $M$  in the model is 0.99.

All the experiments are done in 5 folds to boost the evaluation result stability.

The Adam optimizer is used with the Cross-Entropy loss function. Also, a plateau learning rate scheduler is employed with the configuration of (start=0.01, ratio=0.5, patience=10, cooldown=5, verbose=True). Each model is trained for 150 epochs, and has reached convergence before the last epoch.

### 3.4.3 Rotation Evaluation

An important motivation of this algorithm is to spin a kernel pattern freely. It is crucial to find a method to evaluate the rotation consistency between kernels. We suggest to use a Phase-Offset Correlation(POC) function.

Given two sets of perfectly rotated kernels,  $\mathbf{W}$  and  $\mathbf{W}'$ , not considering the impact of rectangular kernel shape, the correlation between  $(\mathbf{w}_i, \mathbf{w}'_j)$  should always equals to the correlation between  $(\mathbf{w}_{i+k}, \mathbf{w}'_{j+k})$  for any valid  $i, j$ , and  $k$ . In the other words, if kernel set  $W$  is the set to be evaluated, the standard deviation of its correlations with another set of kernels which are precisely rotated reflects the rotational consistency of  $\mathbf{W}$ . If the standard deviation of correlations is equal to 0, then the kernels of all orientations are perfectly rotated. A low POC value suggests that the rotational relation is good. The paradigm sets  $\mathbf{W}'$  is emulated by a set of Gabor kernels. By scanning through all the phase offsets, a more representative conclusion can be drawn. The POC formula is defined as,

$$(\mathbf{w}', \mathbf{g}') = \left( \frac{\mathbf{w}}{\|\mathbf{w}\|}, \frac{\mathbf{g}}{\|\mathbf{g}\|} \right) \quad (3.16)$$

$$\begin{aligned}
\mu_k &= \frac{1}{n} \sum_{i=1}^n \mathbf{w}'_i \cdot \mathbf{g}'_{k+i} \\
POC(\mathbf{W}) &= \frac{1}{n} \sum_{k=1}^n \sqrt{\frac{\sum_{i=1}^n (\mathbf{w}'_i \cdot \mathbf{g}'_{k+i} - \mu_k)^2}{n}} \\
&= \frac{1}{n^{\frac{3}{2}}} \sum_{k=1}^n \sqrt{\sum_{i=1}^n (\mathbf{w}'_i \cdot \mathbf{g}'_{k+i} - \mu_k)^2}
\end{aligned} \tag{3.17}$$

Here, the denominator  $n$  is the number of angles,  $\mathbf{g}$  represents the reference steerable kernel vector emulated by a Gabor function, and  $\mu_k$  is the mean correlation between  $\mathbf{w}$  and  $\mathbf{g}$  given an arbitrary phase offset  $k$ . The standard deviations with different phase offsets are averaged to make the evaluation more representative. One important fact is that POC is not a semantic metric. Being more directionally consistent is not completely equivalent to being better as a semantic director.

Though in Eq.3.16, the scale impact has been eliminated by pattern normalisation, a large kernel that contains more pattern variation has more room for making a mistake. Also, in POC evaluation, Gabor ridge patterns are used as the reference in which the most attention is concentrated on the central part. It can be used to compare the directional consistency when the pattern groups are generally matched. But it will not be very precise if using POC to compare two very different individual patterns.

### 3.4.4 Other Evaluation Metrics

Other regular metrics are also used to evaluate the model, including F-score( $F_1$ ), Sensitivity( $Se$ ), Specificity( $Sp$ ), Accuracy( $Acc$ ), Area Under Curve( $AUC$ ), and G-mean( $G$ ). Their definitions are given as following,

$$F_1 = \frac{TP}{TP + \frac{1}{2}FP + FN} \tag{3.18}$$

$$Se = \frac{TP}{TP + FN} \tag{3.19}$$

$$Sp = \frac{TN}{TN + FP} \tag{3.20}$$

$$Acc = \frac{TP + TN}{TP + FN + TN + FP} \tag{3.21}$$

$$\text{G-mean} = \sqrt{Se \times Sp} \quad (3.22)$$

Here, true positive(TP) represents the true axon pixels that also have a segmentation score greater(less) than 0.5 thus should be classified as positives(negative) class on the map, vice versa. For all the datasets, the number of positive pixels are much greater than the number of the negatives. Though both are important, we treat the sensitivity as more descriptive than specificity in these tasks due to the imbalance fact. It reflects more variation when the architecture is changed. Some collective indicators such as F1, and ROC analysis are also used, as well as G-mean which calculates the geometrical mean of the corresponding sensitivity and specificity. The models' parameter usages are also presented in the tables.

### 3.4.5 General Performance

A MASC model has two main characteristics, the outstanding training efficiency and the kernel rotation feature. In CHASE-DB1 Table.3.1, a MASC 5-2-16 model trained on 6,400 patches is compared with other representative architectures, some of them claimed great parameter efficiency at the times. With fewer than 0.4% of the parameters of the smallest model in the list, our MASC-5-2-16 achieves equivalent result as well as the best F1 score in the group, and has generally outperformed the human baseline. On the other hand, our MASC model shows great balance dealing with positive and negative samples, which only ranked behind U-Net(30M) as the second-best in the G-mean column.

One of the major challenges of vessel tasks is the subtle patterns. However, topologically speaking, successfully detecting these structures is vital to the background of the task otherwise the connectivity of tubes is broken. Considering the examples in Fig. 3.13, MASC-5-2-16 shows its strength in detecting delicate structures, as most thin traces in the pictures are found. At the same time, U-Net and Ladder Net (U-Net variant) failed in these places. For the wide vessels, our model does not give a certain decision as the central parts are lower than the sides. This may be explained by the pre-processing, during which small patches as large as  $48 \times 48$  are cropped. A small patch is not likely to include sufficient context for classifying wide vessels, especially when a random crop seed is located in the middle of them. This knowledge is not learned by the model.

In the DRIVE dataset, a different MASC model is used, in which both the size of

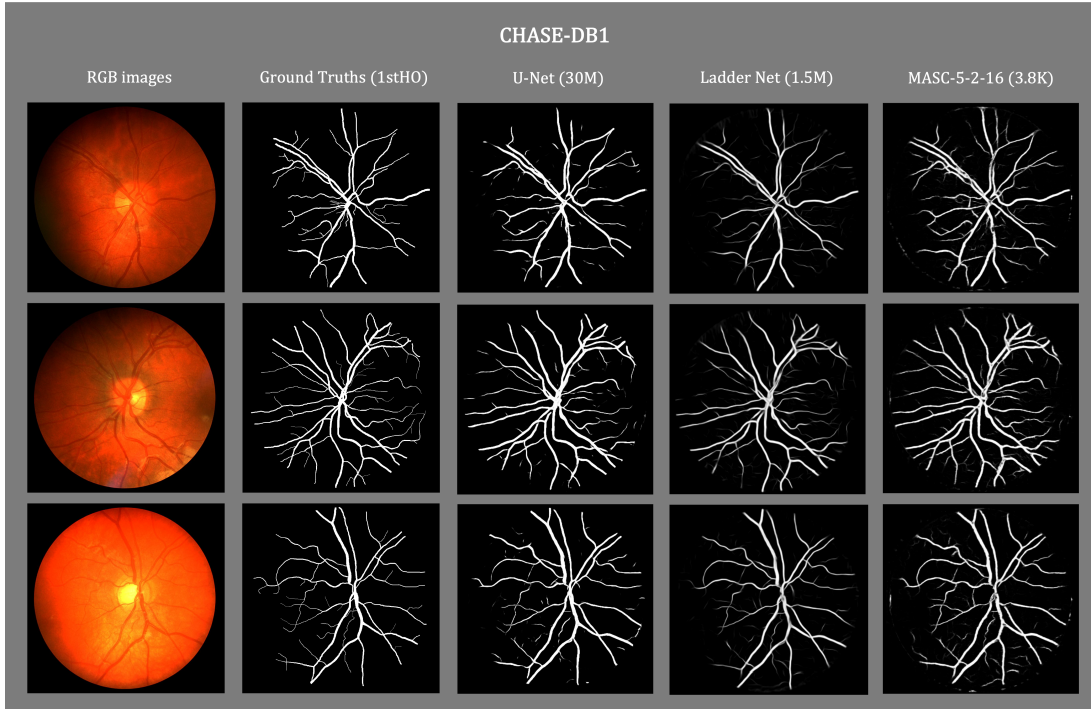


Figure 3.13: Segmentation results from U-Net [101], Ladder Net [148], and MASC-5-2-16 on CHASE-DB1. The parameters of a U-Net model are 20 times that of a Ladder Net model, and are 6,350 times larger than our model. Model MASC-5-2-16 is constructed for demonstration purpose, its performance can be improved using better hyper-parameters.

the vanilla convolutional layers and the MASC units are increased. The dataset contains more tiny vessels, and the vascular center can be on either the left or the right side of an image. Though the total parameter number is raised to 19.6K, the model is still less than 2% of VesselNet in Table.3.1. The methods like MiUNet [55] and IterMiUNet [65] use dense connection and more iterations, thus the computation densities are increased. Just like what we have learned in the CHASE-DB1 experiment, the MASC model is rather sensitive on detecting target patterns, whose Se score is ranked as the second place just behind the LadderNet with a small gap of 0.2%. And the character of good class balance (indicated by G) is found again. The model outperforms the human baseline on all the columns.

### 3.4.6 Hyper-Parameter Studies

MASC unit has several innovations designed to help identify structures. Here we perform experiments to quantify the effects of each one. All the experiments except the

Table 3.1: Segmentation Results on CHASE\_DB1, columns from left to right refers to F1 score, sensitivity, specificity, accuracy, area under curve, geometric mean of sensitivity and specificity, and parameter number of the model. Bold text indicates the largest number in a column.

Methods	F1	Se	Sp	Acc	AUC	G	Params
2nd Observer [95]	76.9	74.3	97.9	94.7	-	85.3	-
U-Net [101]	77.8	82.8	97.0	95.8	97.7	89.6	30.6M
Residual U-Net [141]	78.0	77.3	98.2	95.5	97.8	87.1	7.8M
Reccurent U-Net [2]	78.1	74.6	<b>98.4</b>	96.2	98.0	85.7	1.1M
R2U-Net [2]	79.3	77.6	98.2	<b>96.3</b>	<b>98.1</b>	87.3	1.1M
LadderNet [148]	79.0	78.6	98.0	96.2	97.7	87.8	1.5M
VesselNet [127]	79.1	78.2	98.1	96.2	97.6	87.6	585.8K
MiUNet [55]	74.1	84.4	95.7	95.7	94.7	89.9	69K
IterMiUnet [65]	78.8	<b>84.4</b>	97.0	95.9	98.1	<b>90.5</b>	150K
MASC5-2-16-5 $\times$ 5 (6.4k samples)	<b>80.2</b>	80.7	97.8	95.6	97.4	88.8	<b>3,799</b>

Table 3.2: Segmentation Results on DRIVE

Methods	F1	Se	Sp	Acc	AUC	G	Params
2nd Observer [77]	78.8	77.6	97.3	94.7	-	86.9	-
U-Net [101]	81.4	75.4	98.2	95.3	97.6	86.0	30.6M
Residual U-Net [141]	81.5	77.3	98.2	95.5	97.8	87.1	7.8M
Reccurent U-Net [2]	81.6	77.5	98.2	95.6	97.8	87.2	1.1M
R2U-Net [2]	81.7	77.9	98.1	95.6	97.8	87.4	1.1M
LadderNet [148]	82.1	<b>80.8</b>	97.7	95.5	97.7	<b>88.8</b>	1.5M
VesselNet [127]	81.3	76.3	<b>98.4</b>	95.6	97.7	86.6	585.8K
MiUNet [55]	<b>82.4</b>	79.7	98.0	<b>95.7</b>	<b>98.1</b>	88.4	69K
MASC5-6-16-11 $\times$ 11 (6.4k samples)	81.6	80.6	97.4	95.4	97.5	88.6	<b>19.6K</b>

Table 3.3: Segmentation Results on Axon Dataset(\* from [77])

Methods	F1	Se	Sp	Acc	AUC	G	Params
U-Net [101]	<b>88.0</b>	<b>87.0</b>	<b>99.7</b>	<b>99.5</b>	93.5	<b>93.1</b>	30.6M
MASC5-6-16-11 $\times$ 11 (6.4k samples)	81.6	80.6	97.4	95.4	<b>97.5</b>	88.6	<b>19.6K</b>

Table 3.4: Module Ablation Results

Methods	F1	Se	Sp	Acc	AUC	G	Params
2nd Observer	76.9	74.3	97.9	94.7	-	85.3	-
w/o MASC unit	72.3±0.7	62.8±1.2	<b>98.7±0.1</b>	94.7±0.1	96.0±0.2	78.7±0.7	2,819
replace MAC	76.6±0.5	73.0±1.1	97.8±0.2	95.2±0.1	96.5±0.1	84.5±0.6	3,409
w/o MSC	76.1±0.3	69.6±0.9	98.1±0.1	95.1±0.0	96.4±0.1	82.9±0.5	3,799
replace MASC Units	77.0±0.9	74.0±1.1	97.7±0.1	95.1±0.2	96.3±0.3	85.0±0.6	3,329
replace MASC Blocks	76.9±0.3	70.4±0.8	98.4±0.1	95.3±0.1	96.7±0.1	83.3±0.4	9,009
MASC-5-2-16	<b>78.2±0.4</b>	<b>74.6±1.0</b>	98.0±0.1	<b>95.4±0.0</b>	<b>97.1±0.1</b>	<b>85.5±0.5</b>	3,799

training volume study are trained and tuned under 5 fold cross validation with 4,000 samples, which means 3,200 samples were fed in each run and 800 samples for validation. The test set is kept separately. Their results are presented and discussed.

### 3.4.6.1 Ablation Experiments

In a MASC unit, the rotation character is achieved by leveraging hybrid convolution with response shaping, and the scale character uses pyramid representations. According to the nature of a task, a unit can be separated and used independently. For example, if a pattern only has rotation variance but with almost uniformed breadth, then the scale module should be skipped for faster computation. These decisions require us to know the impact of changing each part. We have four components in this model, Multi-Angle and Scale Convolution block (MASC block), Multi-Angle and Scale Convolution unit (MASC unit), Multi-Angle Convolution unit (MAC unit), and Multi-Scale Convolution unit (MSC unit). A MASC unit refers to a standard unit shown as Fig. 3.12(c) with both rotation and scale invariant features. A MASC block is the building block of this model with several parallel organized MASC units in it, displayed by Fig. 3.12(c). A MAC unit is a combination of oriented hybrid kernels and response shaping module, and an MSC unit refers to the pyramid scale searching structure.

We compared the models with either or both of MAC-MSC modules skipped or replaced by equivalent convolutional substitutions. More details about this experiment are presented in Table.3.4. And it is clear that both the figures and the numbers suggest that the full version made a better segmentation for the vascular objects, and is closest to the ground truths. The model of replace-MASC used as many parameters as the MASC-5-2-16, but did not control the noises well. It is easy to see more small false positive response in the images. For the cases of non-MASC and non-MSC, their Se score is obviously lower than the average level by 11.8% and 5% respectively. The same message can also be read from the examples in Fig. 3.14, where many small

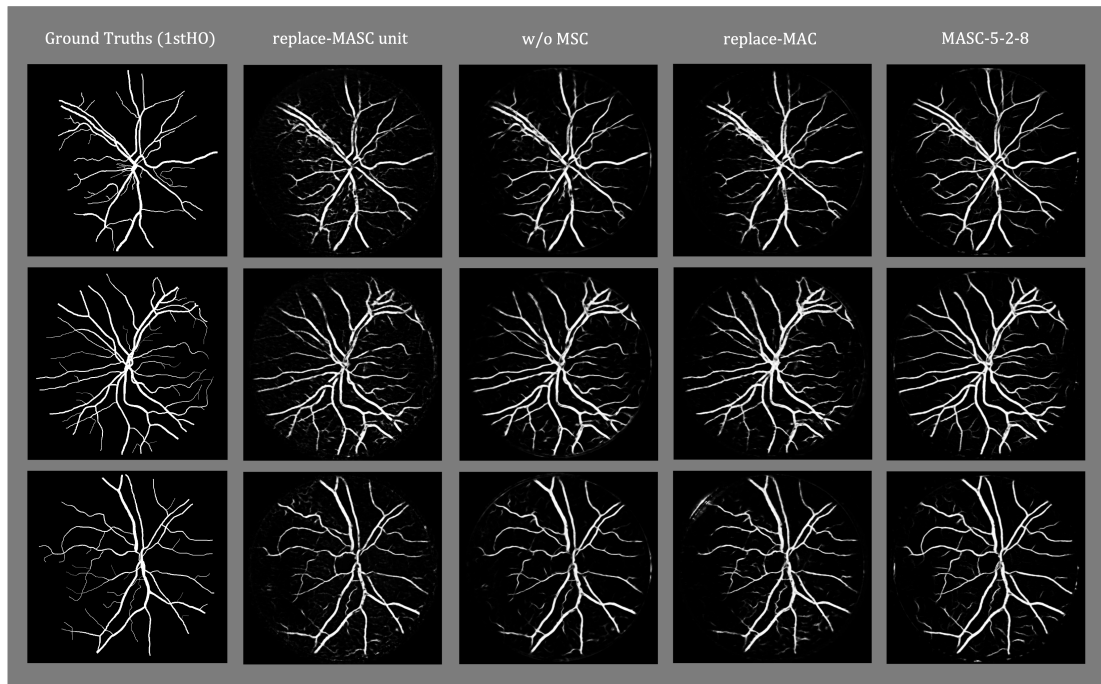


Figure 3.14: Four models are compared with ground truth under the task of CHASE-DB1 vessel segmentation. MASC-5-2-16 is the base model in this figure. Replace-MASC has all MASC units substituted by equivalent normal convolutions in the MASC blocks. And similarly, the replace-MAC model has all the hybrid kernels replaced by normal convolutional kernels with the same input and output channels. The non-MS-C model is the case where the scale searching step is canceled and the rotation searching is executed on the original feature maps.

branches are missed by these two models. And if naively assuming the contribution of MAC is independent of the existence of MSC, the number differences show that removing MAC (Se -6.8%) would cause a worse impact than removing MSC (Se -5%). At the same time, the model that replaces MASC Blocks with 8-8 convolutions used more than doubled parameters but also results in an inferior AUC score and a worse Se&Sp balance than the original version.

#### 3.4.6.2 Parallel MAC Units

More than one MAC kernel can be integrated into a MASC unit. In the example of MASC-5-2-16, we have two MAC kernels arranged in parallel. The inputs of the two MACs are different and their outputs are concatenated. One might expect that adding more MAC kernels to the model will be detrimental to the performance as they are parallel. But the question is if more units are useful. We studied the cases of

Table 3.5: Parallel Units Numbers

Methods	F1	Se	Sp	Acc	AUC	G	Params
2nd Observer	76.9	74.3	97.9	94.7	-	85.3	-
w/o MASC unit	72.3±0.7	62.8±1.2	<b>98.7±0.1</b>	94.7±0.1	96.0±0.2	78.7±0.7	2,819
MASC5-1-16	76.6±0.5	73.0±1.1	97.8±0.2	95.2±0.1	96.5±0.1	84.5±0.6	3,244
MASC5-2-16	78.2±0.4	74.6±1.0	98.0±0.1	95.4±0.0	97.1±0.1	85.5±0.5	3,799
MASC5-3-16	78.5±0.3	75.1±0.7	98.1±0.1	95.5±0.1	97.2±0.1	85.8±0.4	4,354
MASC5-4-16	79.1±0.3	78.4±0.8	97.6±0.2	95.4±0.1	97.3±0.1	87.4±0.4	4,909
MASC5-5-16	<b>79.3±0.3</b>	<b>78.6±0.5</b>	97.6±0.1	<b>95.5±0.1</b>	<b>97.4±0.1</b>	<b>87.6±0.3</b>	5,464

using up to 5 MAC units. The model without MASC unit is treated as the zero-unit case equivalent. Evidence is in Fig. 3.15 where a monotonic upward trend is shared by both sensitivity and F-score when the number of parallel units is increasing. The same conclusion can also be drawn from the Table.3.5. The marginal benefit of using more units is generally decreasing. Also, the indicators are stable when more MAC units are used, as their standard errors are also decreasing, which further confirmed the performance convergence trend.

A possible reason may come from the curvilinear object and the kernel size used, where the oriented patterns do not have that many variations given that detector size. This can also be noticed from the second plot in Fig. 3.15. Though the kernels in block-1 assign different attention to each side, the block is generally learning similar knowledge. One solution is using a larger kernel size so more variations can be included just like the example given in Fig. 3.4. Another reason may come from the dataset, where the model is approaching the ceil of performance.

### 3.4.6.3 Number of Directions

Another question is the choice of direction numbers and re-use operators. The models are separated into two groups, one with a mechanical expansion range of  $2\pi$  and the other is  $\pi$ . By default, a typical MASC model takes  $2\pi$  directional range, which means 4-fold  $90^\circ$  rotations are applied. For  $1\pi$  case, only one time of  $90^\circ$  is used. Comparing MASC5-2-16 with MASC5-2-8( $1\pi$ ) whose parameter numbers are the same, the first one's results are slightly better. In most metrics, MASC-5-2-16( $2\pi$ ) is comparable with MASC5-2-16( $1\pi$ ) whose MASC units use doubled parameters, as well as MASC5-2-32( $2\pi$ ) and MASC5-2-16( $1\pi$ ).

For both  $1\pi$  and  $2\pi$  models, adding more direction intervals do not necessarily improve performance. On the contrary, over-splitting the searching range will lead to worse performance, as the AUC score of MASC 5-2-24( $1\pi$ ) is even lower than



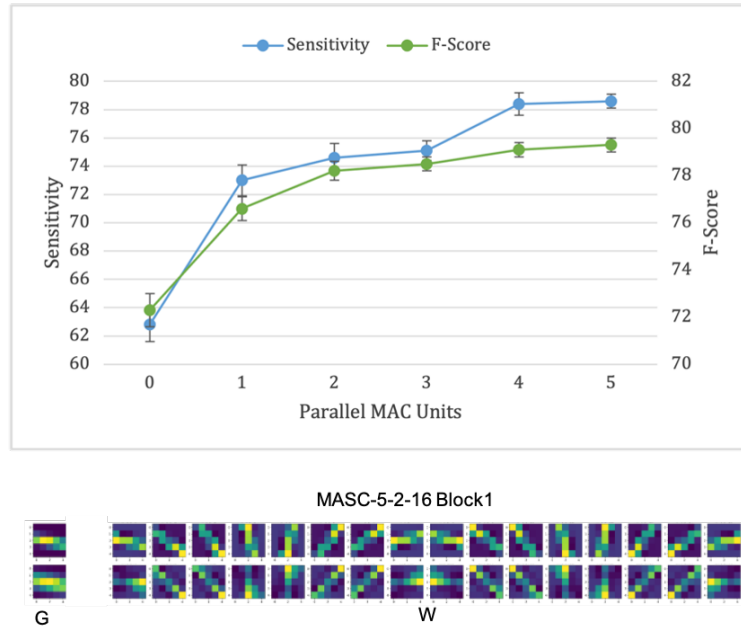


Figure 3.15: Top: Sensitivity and F-score changes using different numbers of MAC units. Bottom: MAC kernel patterns sampled from the block-1 of a MASC-5-2-16 model. The G column exhibits the Gabor bases of the hybrid kernels, and W columns are the hybrid kernels.

MASC5-2-8( $1\pi$ ), though three times as many parameters are used in the same directional detection.

The pattern repetition may exist at the early stage of training  $2\pi$ , but was found able to be eliminated gradually. A good example is Fig.3.6, where the ultimate patterns are not all origin-symmetric, therefore only one peak is presented in the corresponding  $M$  matrix. Different from  $2\pi$  models, an  $1\pi$  model needs to learn a kernel that is closer to origin-symmetric to have a full range coverage. This limits the valid variation space of  $1\pi$  kernels to be only a half of what  $2\pi$  kernels have. Also, to produce a set of 16-directional kernels, a  $2\pi$  MAC unit would have 2 base kernels(before kernel re-use), when a  $1\pi$  MAC unit needs 4. Having more base kernels means taking more uncertainty in updating kernels consistently thus harder to maintain a precise rotational relationship. The significant differences in Table.3.6 also confirmed this point, where the average POC score of 16-directional  $1\pi$  models is two times that of its  $2\pi$  equivalent.

To conclude what we learned about the direction numbers,

1. For vascular objects whose width can vary,  $2\pi$  is a better choice from both accuracy and efficiency considerations.

Table 3.6: MAC Direction Numbers. Mirror kernel re-use is applied by default,  $1\pi$  schemes do not use the mirror transformation only right-angular rotations.

Directions	F1	Se	Sp	Acc	AUC	G	POC ( $10^3$ )	Params
MASC5-2-8	76.7±0.5	73.4±2.0	97.8±0.3	95.1±0.1	96.7±0.0	84.8±1.0	-	3,389
MASC5-2-16	78.2±0.4	74.6±1.0	<b>98.0±0.1</b>	95.4±0.0	<b>97.1±0.1</b>	85.5±0.5	<b>12.9±1.3</b>	3,799
MASC5-2-24	<b>78.6±0.3</b>	<b>75.8±0.4</b>	97.9±0.0	<b>95.4±0.1</b>	97.1±0.2	<b>86.1±0.2</b>	18.3±0.7	4,209
MASC5-2-32	78.1±0.2	75.1±0.6	97.9±0.1	95.4±0.0	97.0±0.1	85.7±0.3	19.3±1.7	4,619
MASC5-2-8( $1\pi$ )	77.4±0.3	74.0±0.4	97.9±0.0	95.2±0.1	96.7±0.2	85.1±0.2	18.6±1.3	3,809
MASC5-2-16( $1\pi$ )	<b>77.9±0.5</b>	<b>74.7±1.4</b>	<b>97.9±0.2</b>	<b>95.3±0.1</b>	<b>97.1±0.1</b>	<b>85.5±0.7</b>	26.4±2.2	4,549
MASC5-2-24( $1\pi$ )	76.8±0.6	73.6±0.5	97.8±0.1	95.1±0.1	96.5±0.2	84.8±0.3	24.9±1.3	5,284

2. A finer orientation search does not always boost performance. On the contrary, too many directional kernels could impact the precision by rendering more direction selection mistakes.
3. Rotation maintenance can be more challenging when using more directions.

#### 3.4.6.4 Directional Kernel Size

Kernel size, especially for directional kernels, is a very critical hyper-parameter that strongly influences model performance. The core presumption in this work is learning rotatable patterns, however, the rotatable facts are different in different scales. For example, a windmill is a windmill only when looking it as a whole. The sampled pattern can be trivial when the kernel size is too small, and may not be rotatable when its size is too large. Although we have used pyramid searching to bridge the contents with different scales, the gap is not completely mitigated. A very straight-forward example to show the importance of kernel size in MASC is the  $9 \times 9$  pattern in Fig. 3.4 and the  $5 \times 5$  pattern in Fig. 3.15. On  $5 \times 5$  level (examples in Fig. 3.7 and Fig. 3.6), even with pyramid searching, the knowledge to be learned is mostly ridge type. What a model sees on  $9 \times 9$  level, such as Fig. 3.4, can include many other forms, which can be circles, wavelets, etc.

As shown in Table.3.7, using a larger kernel size comes with advantages, most columns are improved with a significant margin. However, it also means the parameter usage is increasing significantly.

The POC is not a semantic metric and only reflects the directional consistency. Therefore, a kernel with higher POC is not necessary to be able to act as a better detector, e.g.  $k=5$  and  $k=9$ . Putting the POCs score together, it is obvious that it is harder to keep kernels precisely rotated with a small size such as  $3 \times 3$ . Different

Table 3.7: MAC Kernel Sizes

Methods	F1	Se	Sp	Acc	AUC	G	POC ( $10^3$ )	Params
k=3	74.6±0.3	67.5±0.6	<b>98.3±0.1</b>	94.9±0.0	96.3±0.1	81.5±0.4	42.0±3.0	3,479
k=5	78.2±0.4	74.6±1.0	98.0±0.1	95.4±0.0	97.1±0.1	85.5±0.5	<b>12.9±1.3</b>	3,799
k=9	<b>79.3±0.1</b>	<b>77.7±0.8</b>	97.7±0.1	<b>95.5±0.0</b>	<b>97.5±0.0</b>	<b>87.1±0.4</b>	16.7±3.5	4,919
k=7(F-MAC $\Phi = \frac{\pi}{2}$ )	77.7±0.8	74.6±1.0	97.8±0.2	95.3±0.2	97.1±0.2	85.4±0.5	9.1±0.3	5,929
k=7(F-MAC)	77.8±0.4	73.0±0.8	98.2±0.1	95.4±0.0	97.2±0.1	84.7±0.4	9.0±0.4	5,929

Table 3.8: Key Matrix  $\mathbf{M}$  Momentum

Momentums	F1	Se	Sp	Acc	AUC	G	POC ( $10^3$ )
momentum=0	77.8±0.4	74.4±1.5	97.9±0.2	95.3±0.0	97.0±0.1	85.3±0.8	15.7±1.2
momentum=0.5	76.6±0.5	74.1±1.4	97.6±0.3	95.0±0.1	96.5±0.2	85.0±0.7	15.1±0.8
momentum=0.9	78.2±0.4	<b>75.6±1.1</b>	97.8±0.2	95.3±0.0	97.0±0.1	<b>86.0±0.6</b>	13.0±1.0
momentum=0.99	<b>78.2±0.4</b>	74.6±1.0	<b>98.0±0.1</b>	<b>95.4±0.0</b>	<b>97.1±0.1</b>	85.5±0.5	<b>12.9±1.3</b>

kernel sizes can be used to alleviate the fractional scale gap introduced by pyramid downsampling.

### 3.4.6.5 Response Shaping Momentum

The key matrix  $\mathbf{M}$  in response shaping is updated with a certain degree of momentum. We have found it can effectively stabilise the training process without sacrificing accuracy. Moreover, we noticed a rather large momentum degree can help improve the modeling performance on average. The plus-minus ranges in Table.3.8 are estimated by the standard error(SEM). The smallest SEMs (bold) are mostly located on the bottom rows in which 0.9 and 0.99 are used. Also, with all the other hyper-parameters to be the same, the POC of  $\mathbf{m} = 0.99$  is significantly lower than the case of  $\mathbf{m} = 0$  and  $\mathbf{m} = 0.5$ . In the other words, updating  $\mathbf{M}$  with momentum can contribute to a smooth transaction of directional patterns.

Another interesting fact that may worth attention is how POC changes against epochs given different momentum degrees. An example is shown in Fig. 3.16, in which the POC curves are compared to the Binary Cross-Entropy(BCE) semantic cost chart. Initialised as precisely rotated Gabor patterns, the POC scores started from rather small values and climbed rapidly in the first 10-20 epochs, and then reached a plateau stage. On the other side, the BCE scores dropped sharply with a very similar but reversed pace. A rational presumption is that the new knowledge makes the convolutional layers before and after the MASC units change dramatically at beginning. Together with the strong gradients, all the factors force the directional kernels to respond. Three of the four curves reached their peak POC value soon after the rapid growth stage, and

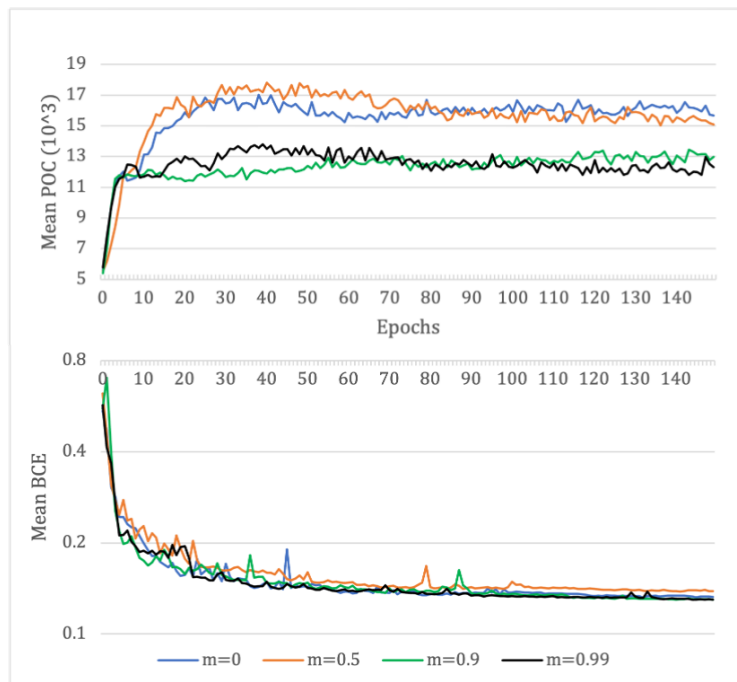


Figure 3.16: The trends of averaged POC (top) and BCE semantic scores (bottom) against training epochs with different degrees of momentum were used. A smaller value is desired under both metrics. Four momentum degrees  $\mathbf{m} \in \{0, 0.5, 0.9, 0.99\}$  are tested. Greater momentum indicates smaller portion of key matrix  $M$  changed every step.

the following slow adjustment stage moderately decreases the in-consistency level till the end of training.

Also, it is surprising to see that  $\mathbf{m} = 0.9$  and  $\mathbf{m} = 0.99$  reached the plateau stage earlier than  $\mathbf{m} = 0$  and  $\mathbf{m} = 0.5$ . In Fig. 3.16 BCE chart, the values which represent the lower momentums are greater than the high momentums between 10-20 epochs. Intuitively, a smaller momentum means updating  $\mathbf{M}$  faster so the directional kernels can adapt to the changes swiftly. But the experiment data shows actually it causes a model to learn slower. Similar conclusions were drawn in the contrast learning experiments where momentum mechanism is used in updating crude models. In our work, momentum is designed to avoid fast and early distraction in  $\mathbf{M}$  where the model can have a chance to gradually shift back.

#### 3.4.6.6 Kernel Construction and Initialisation

Directly deciding the initial shape of key matrix  $\mathbf{M}$ , the initialisation of the hybrid kernels, especially the free updatable part  $\mathbf{C}$ , would have a profound influence on the directional kernels. Regarding this question, we tested three different approaches, defined as,

$$\text{Type 1: } \mathbf{W} = \text{Gabor}(\mathbf{C}) \quad (3.23)$$

$$\text{Type 2: } \mathbf{W} = \mathbf{G} \odot \text{Ones}(\mathbf{C}) \quad (3.24)$$

$$\text{Type 3: } \mathbf{W} = \mathbf{G} \odot \text{Gabor}(\mathbf{C}) \quad (3.25)$$

The functions  $\text{Gabor}(\mathbf{C})$  and  $\text{Ones}(\mathbf{C})$  represent the two ways of initialising the flexibly updatable part  $\mathbf{C}$  in a hybrid kernel, with Gabor+noise weights or one+noise weights respectively. Type-1 models are different, in which the hybrid structure is substituted by a normal convolutional node shaped like a Gabor pattern. A type-3 pattern is a second-ordered Gabor. The kernels of all three types are normalised after initialisation and their outputs are managed by response shaping. The experiment results are presented in Table.3.9. There are some differences in the columns such as F1, where type-3 has some advantages. But generally speaking, the gap is not significant. However, the models can be discriminated by their rotation abilities. The type-3 models have a much lower POC than Type-2 and Type-1. This makes its performance variance also narrower, which is a desired property.

Table 3.9: Kernel Construction and Initialisation

Methods	F1	Se	Sp	Acc	AUC	G	POC ( $10^3$ )
w/o MASC unit	72.3±0.7	62.8±1.2	98.7±0.1	94.7±0.1	96.0±0.2	78.7±0.7	-
replace MAC	76.6±0.5	73.0±1.1	97.8±0.2	95.2±0.1	96.5±0.1	84.5±0.6	-
MASC(type1)	77.8±0.7	74.3±1.5	98.0±0.3	95.4±0.2	97.1±0.3	85.3±0.8	30.4±1.9
MASC(type2)	77.4±0.4	75.0± <b>0.7</b>	97.7±0.1	95.2±0.1	96.8±0.1	85.5± <b>0.4</b>	27.3±1.1
MASC(type3)	78.2± <b>0.4</b>	74.6±1.0	98.0± <b>0.1</b>	95.4± <b>0.0</b>	97.1± <b>0.1</b>	85.5±0.5	<b>12.9</b> ±1.3

Compare to element **C**, element **G** is more important to the hybrid kernel initialisation. In the pilot experiments, we found it is critical to distribute the directional kernels sparsely in the weight space, so the chance of two kernels learning from the same orientation can be controlled. A simple solution to generate low-correlated pattern is initialising  $\gamma$  (see Eq.3.4) to be high. Gabor function can be understood as expanding a wavelet function under Gaussian distribution on a 2D plane(demonstrated in Fig. 3.3 (a)). The parameter  $\gamma$  controls the  $x/y$  variance ratio in the function. A pattern will be narrow and concentrate on the central part when it is high, thus decreasing the correlation between two directional patterns as the overlap is small. Usually, we set the variance parameter  $\sigma$  on  $x$  direction to be a quarter of kernel size,  $\lambda$  to be double kernel size, and initialise  $\gamma \in (6, 12)$ . An illustration example of the initial pattern is on the last row of Fig. 3.3 (b).

### 3.4.6.7 Message Integration Methods

As suggested by [53], we tested several kinds of message integration method, including response shaping (see  $R(\mathbf{v})$  in Eq.3.8, Eq.3.11), maximum response, and squared response difference. And the latter two definitions are given as,

1. Max response:

$$\Phi(\mathbf{v}) = \max(\mathbf{v}) \quad (3.26)$$

2. Squared Response Difference, given  $\|\mathbf{m}_i\| = 1$ :

$$\begin{aligned}
\Phi(\mathbf{v}) &= \min_i(\|\mathbf{m}_i - \mathbf{v}_i\|^2) \\
&= \max_i(2\|\mathbf{m}_i\| \times \|\mathbf{v}_i\|R(\mathbf{v}_i) - \|\mathbf{m}_i\|^2 - \|\mathbf{v}_i\|^2) \\
&= \max_i(2\|\mathbf{v}_i\|R(\mathbf{v}_i) - 1 - \|\mathbf{v}_i\|^2) \\
&= 2\max_i(\|\mathbf{v}_i\|(R(\mathbf{v}_i) - \frac{1}{2}\|\mathbf{v}_i\|)) - 1
\end{aligned} \quad (3.27)$$

Variable  $\mathbf{v}$  which is the encoded directional response vector has been defined in

Table 3.10: Response Shaping, Max Response, and Squared Response Difference

Methods	F1	Se	Sp	Acc	AUC	G	POC ( $10^3$ )
replace MAC	76.6±0.5	73.0±1.1	97.8±0.2	95.2±0.1	96.5±0.1	84.5±0.6	-
Max Response	77.2±0.3	73.5±0.3	97.9±0.1	95.2±0.0	96.7±0.1	84.8±0.5	24.4±1.1
Squared Response Difference	76.8±0.9	72.1±2.4	98.1±0.2	95.2±0.1	96.7±0.1	84.0±1.3	23.7±2.2
Response Shaping	<b>78.2±0.4</b>	<b>74.6±1.0</b>	98.0±0.1	<b>95.4±0.0</b>	<b>97.1±0.1</b>	<b>85.5±0.5</b>	<b>12.9±1.3</b>

Eq.3.5. We chose response shaping which is multiplication based as it has the best AUC performance and intra-category balance. Also, it has the most smooth directional evolution among the three as its POC score is also outstanding in Table.3.10. All the methods outperformed the replace-MAC models which are equivalent to the case without message integration involved.

Given  $\mathbf{m}$  normalised,  $\|\mathbf{m}\| = 1$ , the equation of squared difference can be simplified to a scaled and shifted version of response shaping. Though, response shaping is still a more descriptive function than squared difference. Two input vectors that have the same Euclidean distance to  $\mathbf{m}_i$  may have completely different response shapes. Both methods suppress the signals from close directions and enlarge the signals from far directions.

A good message integration module can bring improvement to the task and also help maintain the directional links. To our best knowledge at the time of writing, response shaping is the optimal tested option. The method has a theoretical connection with attention mechanism, and the new findings in the area will be updated to the MASC model in our future work.

### 3.4.6.8 Training Volume

Another aspect of modelling efficiency of MASC is it requires far fewer training samples. The illustration is give in Fig.3.17. With 160, 400, 800, 1,600, 3,200, 6,400 training samples, our MASC-5-2-16 model outperformed the U-Net baseline in the 5-fold test. If evaluate the model by AUC, to achieve the same performance of the baseline’s 6,400 level, a MASC model only needs fewer than 800  $48 \times 48$  image patches, which is less than 12.5% of the U-Net requirement. And if evaluate with ACC, to reach the manual annotator’s accuracy, MASC will need 800 samples by average, when a U-Net takes 1,600 samples.

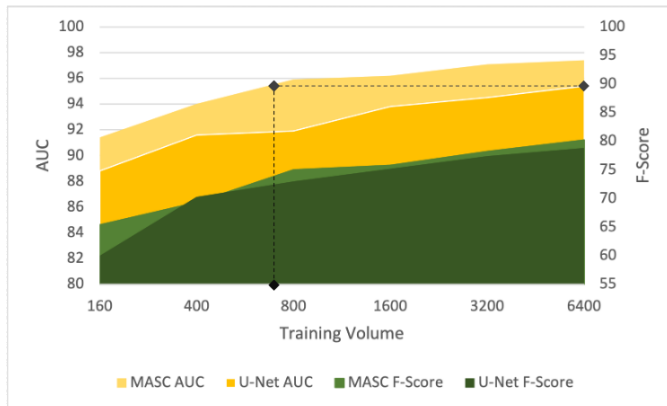


Figure 3.17: A comparison between MASC-5-2-16 and a U-Net baseline with different numbers of training samples fed. The dash line indicates the required training samples for a MASC-5-2-16 to achieve the same AUC score of an average U-Net model that is trained with 6,400 samples. The example U-Net architecture(573k parameters) is 121 times larger than the MASC model. The plot shows MASC model is more efficient on both necessary parameter number and training samples.

Table 3.11: Training MASC-5-2-16( $5 \times 5$ , 3,799 parameters) and U-Net (573k parameters) with Different Sample Volume

Methods	Samples	F1	Se	Sp	Acc	AUC	G
2nd Observer	-	76.9	74.3	97.9	94.7	-	85.3
U-Net	160	60.0±3.0	54.5±2.9	96.5±0.6	91.9±0.7	88.8±1.0	72.4±2.0
	400	70.3±0.4	64.9±1.1	97.5±0.2	93.9±0.1	91.6±0.3	79.6±0.6
	800	73.0±0.4	67.7±0.8	97.8±0.1	94.5±0.1	91.9±0.4	81.4±0.5
	1,600	75.2±0.9	73.2±2.0	97.3±0.2	94.7±0.1	93.8±0.5	84.4±1.1
	3,200	77.4±0.7	76.5±1.1	97.4±0.1	95.1±0.1	94.5±0.4	86.3±0.6
	6,400	78.8±0.1	79.7±0.6	97.2±0.1	95.3±0.0	95.4±0.2	88.0±0.3
MASC	160	65.4±2.4	61.3±3.0	96.8±0.4	92.8±0.5	91.4±1.0	76.9±1.9
	400	69.3±1.5	61.9±2.7	98.0±0.1	94.0±0.2	94.0±0.4	77.8±1.6
	800	75.0±0.2	72.3±0.8	97.5±0.1	94.7±0.1	95.9±0.1	83.9±0.4
	1,600	75.8±0.7	71.5±1.2	97.9±0.1	95.0±0.1	96.2±0.3	83.6±0.7
	3,200	78.2±0.4	74.6±1.0	98.0±0.1	95.4±0.0	97.1±0.1	85.5±0.5
	6,400	80.2±0.3	80.7±0.9	97.8±0.1	95.6±0.1	97.4±0.1	88.8±0.5



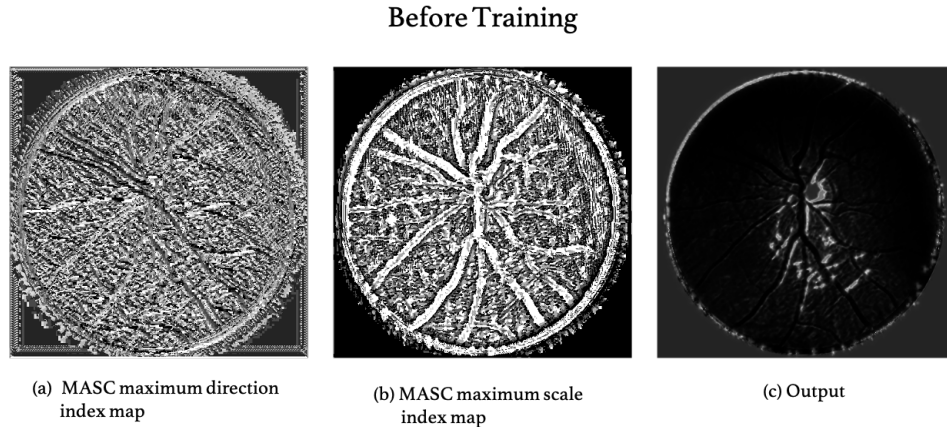


Figure 3.18: The maximum response direction and scale index map generated by a MASC-5-2-8 model before training. The MASC units are expected to generate some semantic relevant feature as the kernels are initialised with a Gabor function which is used in edge detection. But the feature maps are processed by some crude convolutional layers before the MASC unit, during which the information may have been disturbed. The plot (c) is the direct output of the unit.

## 3.5 Discussion

### 3.5.1 Index Maps

We have displayed how a standard MASC model would perform in a standard situation. However, when the data is extremely limited, the back-propagation gradient is not functioning as expected, or a model working poorly for unknown reasons. The structure of MASC enables us to understand why it is not working properly by checking the index maps.

A unique property of MASC is it has semantically comparable kernels. When applied to oriented or scale-variant patterns, a MASC unit makes selections considering the input information not necessary to have much knowledge about it. There is a critical latent coherence in its decisions. Even for a completely irrelevant pattern, a MASC unit that was just initialised but not trained at all still knows which direction and scale it best matched. This may not make sense for an individual pixel. When it come to image level, this semantically coherent decision-making procedure is not fully relying on gradient to optimise. Generally speaking, it is unsupervised to some degree. An example is displayed in Fig. 3.18.

A new model that learning from index maps and uses POC to assist kernel rotation

is introduced in Chapter 5. It is developed for solving more general rotation and scale-invariant cases with even better accuracy and modelling efficiency.

### 3.5.2 Memory Optimisation Choice

To execute best-matched scale selection, a MASC model is required to hold all the copies of feature maps both before and after orientation searching. Though feature channels are compressed to a small number before MAC unit, the memory requirements after scale duplication are increased. This pressure can be optimised via task scheduling.

The straight-forward way of picking maximum sends all the scale tasks in parallel to the GPU. This scheme is time efficient. By turning parallel processing to serial processing, the time advantage can be converted to memory efficiency. As shown in 3.9, the only change is comparing feature maps immediately after MAC unit. In this case, only two copies are held at the same time.

## 3.6 Conclusion

We have introduced a novel model for analysing curvilinear structures which are composed of self-similar elements at arbitrary orientation and scale. The system learns a set of filters that can be transformed easily to produce responses at a range of angles. We show how this can be extended to include a range of scales. The resulting model is very parameter efficient.

The MASC approach can also be thought as a type of attention mechanism. The directional response vector  $v$  can be regarded as the query vector, and the optimal response matrix  $M$  can be treated as the keys. Comparing to ViT (vision transformer) [79, 140] on ImageNet, a medical task typically does not contain arbitrary objects, and useful features tend to be local. Unlike the common self-attention unit, here, the encoder for query and key are the same, and the model has  $n$  candidate keys for each query. Also, it does not refer to context information to compute keys but uses a learned bank of patterns. Since the mechanism is different from the self-attention, in the paper, we named it as response shaping.

The problem of curvilinear segmentation is clearly defined. We are testing our

design to make sufficient performance with least model complexity (parameter number). Four factors in this design contribute to this efficiency, hybrid kernel, rotation-equivalent kernel group, kernel re-use, and scale-axis searching. Giving  $n$  kinds of pattern with  $a$  orientations,  $s$  scales, and mixed with  $k$  kinds of deformations, a MASC model can address the task with  $\frac{a \times n \times k}{4}$  kernels. On the task of retina vessel segmentation, the model achieves accuracy equivalent to the benchmark U-Net variants with only 0.4% of the parameters and 12.5% of the training set. It is thus potentially very useful when only limited numbers of training examples are available. Though this work focuses on retinal images, we have also applied it successfully to tracking growing axons in microscopy images.

Including re-used kernels, the constrained basis in hybrid kernel, and response shaping, there are three factors that have been introduced in this chapter to encourage the directional kernels to rotate consistently.

# Chapter 4

## General MASC

### 4.1 Introduction

In Chapter 3, we introduced a framework for rotation and scale-invariant detection, which is called MASC. Kernels are managed in equivalent sets and updated via gradient descent to preserve rotational symmetry and learn semantic knowledge.

The method is different from normal convolutional neural networks which use large numbers of parameters with augmented data to cover rotational variants. Such an approach inevitably leads to redundancy. The experiments have proven that MASC structure can drastically improve modelling efficiency for the given type of task.

The original curvilinear MASC has encouraging results. Its kernels are linked, looking for comparable features (e.g. orientations). These comparable features enable a model to process features by property rather than follow a full connection between channels.

In this framework, **property** refers to a pattern's orientation and scale, but should not be limited to these. A model looking into properties like MASC would have better interpretability, and it is hoped can be a useful pathway for general artificial intelligence. For example, if a self-driving model can be aware of the scale and moving direction of an object, then it can make decisions regarding the distance to this object. Not necessarily as part of additional processing as [91] described. In another example, [121] leverages tubular scale information from a module model to help biological structure segmentation. These operations can be simplified under the framework of MASC.

To achieve this objective, it is important to enhance the robustness of property detection in MASC. This is the major innovation of the new algorithm.

Generalisation is another question that has not been answered in the original MASC, because the model was established for a particular task. Its kernels are initialised to be similar to the target, which were ridge like structures, using Gabor initialisation. This helps the model to overcome some latent problems of response shaping which have been described in Sec.3.2.4. If we were to use a fully random initialisation scheme, the rotation symmetry preservation weakness can be enlarged, resulting in a less accurate prediction and kernels in a set may not be rotated versions of one another. The ultimate goal is to make MASC units suitable for application to a wide range of features occurring in images.

In this case, more reliable directional kernel sets  $\mathbf{W}$  and key matrices  $\mathbf{M}$  are required. This should not impede the new algorithm from being fully differentiable.

We have demonstrated how to read the rotational symmetry from visualised  $\mathbf{M}$  and introduced the PoC error. A well-rotated model usually has its  $\mathbf{M}$  matrices diagonal-aligned and its PoC error in a reasonable range. We mentioned them here because these factors can actually be transferred to rotation supervising tools.

All these factors together carved out a generalisation solution, a reliable sense of direction for neural networks in general conditions.

We named the new algorithm as General MASC (G-MASC). Compared with the previous version, G-MASC has three major additions, a new rotation penalty called Phase-offset Rotational Error (PoRE), an improved calibrated response shaping function, and a new algorithm for encoding local orientation and scale index maps. The computation complexity and model size are increased in G-MASC, compared to the original MASC, but are still lower than any benchmark methods (0.5% of R2U-UNet).

We are aware that generalisation could mean losing specificity and then efficiency in some conditions. This chapter will summarise the advantages and disadvantages of the additions and give insights into the design choices.

## 4.2 Motivations

Here we explain why we expand upon the MASC approach.

### 4.2.1 Feasibility Discussion

When trained on tasks more complicated than ridge detection, it was found that MASC kernels are less easily interpreted and don't necessarily retain rotationally consistent sets. Nevertheless, on the original MASC, we also noticed that a ridge-initialised kernel can have a range of structures after training. This grants us some confidence, suggesting that it is possible to learn complex patterns from a steerable initial (plus random variation) using kernel reuse and response shaping. Some examples are presented in Fig. 4.1 (a).

Another interesting observation is that the capability of a kernel to preserve rotational symmetry is closely related to its size. Given a convolutional kernel of  $5 \times 5$  and a retina dataset, the trained detectors are mostly found to have a ridge-alike form. If a MASC model's kernel size is increased to  $9 \times 9$  or greater, with the same training set, the ultimate kernel patterns are more diverse. However, a larger kernel would generally increase the PoC error because more uncertainty is involved and it is harder to keep consistent orientations.

From the scale indexes shown in Fig. 4.1 (b), we learned that two kernel sets can be sensitive to targets on different scales. The scale gap between them can be as large as a factor of four. This means the target for one detector is a  $k \times k$  object when the other one is looking for a  $4k \times 4k$  object. Basically, they are learning completely different knowledge.

The three findings above make us realised that it is possible to use MASC to model general patterns.

### 4.2.2 Challenges

Based on the original framework, we summarised the challenges towards a generalisation solution as follows.

1. **Pattern Symmetry:** The target pattern learnt by a MASC kernel was assumed to have line symmetry.

The assumption has to be updated, as it only suits a small portion of cases. Images such as cells and organs are not in this category.

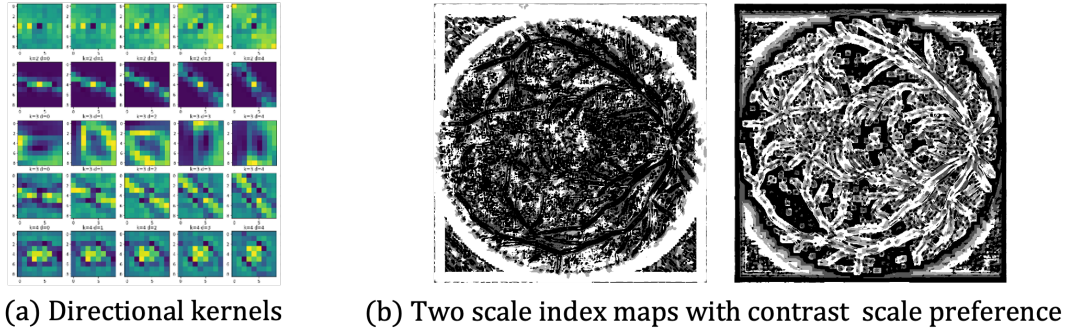


Figure 4.1: In the index map, the value indicates the scale of an interested pattern. There are four scales used,  $\{1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}\}$ , represented by deep to light colours. For the same area, the two kernel sets from one MASC unit give response to the features on different scales. The receptive area of the kernels that generated the left map is four times larger than the one that produced the right map.

This change makes a  $\pi$  direction range for  $\mathbf{W}$  not sufficient anymore. The searching has to be conducted in  $2\pi$  space. Besides that, the transpose action (reflection) will be kept as a kind of new transformation but not as a rotation analogy.

For vessel images, it was appropriate to set the Gabor part in a hybrid kernel that always has zero phase offset. This should be flexible in a general model.

2. **Kernel Construction:** A hybrid MASC kernel is initialised by a random Gabor function.

A G-MASC model needs to involve more variation. Initialising a hybrid MASC kernel is not as easy as it for a normal network, as the randomness has to be rotation-aligned on both parts of it.

In the previous curvilinear experiments, the models are infused with the prior knowledge of target shapes. For general image modelling, a CNN can use prior knowledge via transfer learning. This is not valid for MASC, which requires oriented copies for all the base kernels.

Generating continuous rotational expression for an arbitrary pattern is very useful for both scenarios.

3. **Calibrated Expectations:** Response shaping manages an object's directional response expectations by its key matrix  $\mathbf{M}$ .

The function encourages a kernel to move towards a target pattern following the guidance from the most matched response shape vector, which also downweights

the unmatched ones.

We cannot directly estimate the ultimate response expectations. The initial  $\mathbf{M}$  for a general pattern can be very different from its ultimate form. This requires  $\mathbf{M}$  to be more adaptive and to be updated rapidly in a stable way, which can be achieved by decreasing the batch size, increasing the momentum coefficient, and calibrating the expectation in forward propagation.

In the curvilinear image task, the averaged evolution trajectory in kernel space is shorter than the necessary distance for a common task, as random seeds have been designated to close positions around targets at the beginning. In G-MASC, as mentioned, we will still initialise kernels as precisely rotated. But in order to overcome the increased uncertainty, stricter rotation symmetry protection will be applied.

### 4.2.3 Rotation Symmetry Supervision

A strong signal is produced when orientational feature vector  $\mathbf{v}$  matches a row in  $\mathbf{M}$ . A MASC unit learns the target pattern by solving  $n$  equations.

$$\begin{aligned} \max R(\mathbf{M}_i, \mathbf{v}) &\rightarrow \mathbf{M}_i = \mathbf{v}_{\text{target}} \\ &\rightarrow \mathbf{M}_{ij} = \mathbf{W}_j \cdot \mathbf{X}_{\text{target}} \end{aligned} \quad (4.1)$$

Shown in Eq.4.1, directional responses  $\mathbf{v}_j$  are calculated by convolving a directional kernel  $\mathbf{W}_j$  with input features. And this output is maximised when the directional kernel is as same as the target pattern, as  $\mathbf{W}_j = \mathbf{X}_{\text{target}}$ .

Here,  $n$  as the hyper-parameter for direction number can be any of  $8 * k (k \in \mathbb{N})$ , which was normally set to be 16. Too many directions will encourage the consecutive kernels to stick to each other and influence the precision of the later direction selection step. In Sec.3.4.6.3, the experiment results have shown that the accuracies are similar with different  $n$ , but the computational cost rises linearly.

The size of a directional kernel is the dimension of the corresponding response function. There are a number of  $k^2$  parameters in a base kernel trained with  $n$  constraints defined in  $\mathbf{M}$ . This process would face a  $n < d$  underdetermined condition if the kernel size is too large and a  $n > d$  overdetermined condition if it is very small. The response shaping function is not solved analytically but iteratively which alleviates the problem to some degree, it is still a fundamental issue of delivering sensible detection.



Here are some examples to help understand how this problem could impact detection.

Evidence is from the kernel size experiments in Chapter 3 (see Tab.3.7). In the table, the PoC of the models with  $k = 3$  ( $n > d$ ) and  $k = 9$  ( $n < d$ ) are significantly higher than the model with  $k = 5$ . A high PoC indicates a broken rotation symmetry within a kernel set.

In the controlled direction number experiments (Tab.3.6), the  $2\pi$  MASC model employed twice as many directions which has a much lower PoC than the  $1\pi$  counterpart. In the experiment of increasing the number of directions together with the number of base kernels, the PoC error rises more moderately. Though the  $2\pi$  rows with the same number of base kernels are still lower than the  $1\pi$  rows.

Based on these comparisons, we have learned it is better to use a more multiplicable kernel reuse scheme since it covers both issues. This means the transpose kernel reuse should be kept in G-MASC even under the asymmetric assumption.

A sufficient large kernel size is preferred, the conflict between  $n$  and  $d$  has three common solutions,

1. Impose regularisation to enhance weight correlation;
2. Decrease the problem dimension;
3. Increase the number of constraints.

The common regularisation function based on the  $L_2$  norm of  $\mathbf{w}$  does not work for G-MASC. Since all the directional kernels would be normalised every time forward propagation, its effect is neutralised.

One can decrease the dimension by choosing a smaller kernel size. However, we have learned from the previous experiments that kernel size is an important factor for MASC models and has to be above a minimum level to ensure its normal function.

The last option is to solve the problem by adding more constraints. This can be achieved from two angles. One is to maximise the reuse degree, which can directly put more constraints on each kernel during response shaping. Second, external constraints can be defined in a penalty function. This function is inspired by the PoC and the multidimensional scaling function (MDS). It is calculated as a phase-offset mean squared residual, and named as Phase-offset Rotational Error (PoRE).

The number of additional constraints should be able to fill the gap, because the stresses from response shaping and PoRE perform differently. The choice of this new hyperparameter requires tests to be confirmed.

In conclusion, the modifications of G-MASC unit take three aspects.

1. Increase the kernel reuse degree.
2. Enable more variation in kernel initialisation.
3. Supervise kernel rotation symmetry in the cost function.

#### 4.2.4 Direction and Scale

Assuming a directional kernel set are well aligned, for any oriented pattern the response shaping function is able to generate an orientation label at each position, which is the index of the best-matched row in  $\mathbf{M}$ . Here, an orientation is defined in a relative manner. For example, a  $30^\circ$  object is realised by comparing it to the  $15^\circ$  and  $45^\circ$  cases. Moreover, the pyramid searching function also returns scale labels.

To summarise, a label is associated with a particular object in an orientation and a scale and will make sense only when context labels are given. These labels are the byproducts of G-MASC which do not take additional computation to acquire. The next question is how to use them.

To answer this, our idea is to turn properties into features. Like the positional encoding in vision transformer [31] and relative context networks [53], we believe orientation and scale labels can also be encoded and concatenated to the feature flows. We call this procedure Index Map Featurisation (IMF).

The procedure for different kinds of index maps should also be different.

Starting from orientation labels, some visualised examples have been illustrated in Fig.3.18. For a straight vessel, its label distribution is typically concentrated. When a vessel is making a turn, the local label variance would increase. The target area labels have regular distributions, whose mean value may change. In contrast, the labels for background areas are more random, but generally follow a constant distribution with estimable mean and variance.

Convolution will be used as the encoding method, which is a local function. Removing patch means can make the inputs orientation-calibrated. As the experiment is on isotropic objects, the task's segmentation quality will not be influenced if discarding the absolute orientation information. This will help simplify the task as an orientational IMF module only needs to deal with the relative label pattern.

Another thing that needs to be clarified is how the reflection kernel reuse is addressed in orientational IMF. Reflection will not be used as an analogy of rotation, so

the reflected and the non-reflected labels are arranged separately.

In this project, neural cells live on a plate with a fixed camera distance. The objects belonging to the same category would have close scales. Similarly, MASC units would have a major scale during pyramid representation and output a most frequent label.

Different scale labels usually indicate different objects. The scale index map encoding is like spectral analysis. Labels are sorted into different channels.

Because property label collecting is not differentiable, these index maps should be interpreted as intermediate inputs and only the proceeding kernels will be impacted.

The value of featurised index maps depends on the application scenarios. For a self-driving task, scale IMF can help model making decisions based on the distance changes which can be read from object scales. For some other tasks, some properties may be irrelevant or less useful.

## 4.3 Algorithm

The model of G-MASC is developed from the original MASC in Chapter 3. This section will focus on the new features, the two modifications and the two new modules. The modifications are the new kernel scheme and the calibrated response shaping. The new modules are the rotational penalty PoRE and the index map featurisation modules.

The following descriptions make some changes to the use of terminology.

A base kernel and its variants generated via kernel reuse are called a **directional kernel group**.

A **directional kernel set** refers to all the directional groups affiliated to a MAC unit. In some cases, the term **directional kernel set** is interchangeable with **G-MAC unit**.

For example, we have a G-MASC block. This block has a G-MASC unit that employs 5 parallel G-MAC units. Each G-MAC unit searches in 16 directions. Let us say the reuse multiplication degree is adjusted to the maximum level which is 8. Then, it can be known that this G-MAC unit or this directional kernel set has 2 base kernels and is composited by 2 directional groups with a total of 16 directional kernels.

The concepts arranged in ascending order are directional kernel, directional kernel group, directional kernel set(G-MAC unit), G-MASC unit, G-MASC block.

### 4.3.1 Kernel Scheme

Suppose  $\mathbf{w}$  is a  $k \times k$  kernel at orientation  $\theta$ . Rotating it for three times can give us four variants as  $\{\theta, \frac{\pi}{2} + \theta, \pi + \theta, \frac{3\pi}{2} + \theta\}$ . Transpose them, we will get another set of four reflected kernels.

Similarly, in the case of  $n = 16$ , a MAC set would have two such base kernels at  $\theta = \{\frac{\pi}{8}, \frac{3}{8}\pi\}$  respectively, leading to 8 rotation variants and 8 reflection variants. Each base kernel is a Hadamard product of a trainable Gabor function  $\mathbf{g}_\theta$  and an identically initialised free kernel, written as  $\mathbf{g}_\theta \odot \mathbf{c}_\theta$ . A kernel is normalised, so that  $\mathbf{w}_\theta = \frac{\mathbf{g}_\theta \odot \mathbf{c}_\theta}{|\mathbf{g}_\theta \odot \mathbf{c}_\theta|}$ .

The re-use degree in an equivalent directional kernel group is increased to the maximum level. So that with the same directional granularity, the number of base kernels is minimised. The most significant factor in rotation symmetry is intra-group consistency. Fewer base kernels and kernel groups make its maintenance easier during training.

On another hand, if the number of base directional kernels (before reuse) is fixed, using as many as possible directional detectors can strengthen the collective response

shaping process.

And a small phase offset  $\phi$  as demonstrated in Eq.3.4 is enabled in Gabor initialisation to discriminate a reflection from a rotation at a particular orientation. So they will not confuse the shape selection in response shaping.

### 4.3.2 Calibrated Response Shaping

The approach of constructing a key matrix  $\mathbf{M}$  has been introduced. It provides an adaptive reference for estimating response expectations. There can be many occasions in which  $\mathbf{M}$  is gradually biased, and passing a wrong message to the directional kernels. When using a biased  $\mathbf{M}$ , the intra-group inconsistency can be further magnified, which will deteriorate the modelling quality at the end.

To encourage directional kernels to have  $\frac{4\pi}{n}$  even intervals,  $\hat{\mathbf{M}}$  which is also a distance estimation is better to be a circulant matrix, such that each row is a shifted version of the next,  $M(i+1, j+1) = M(i, j)$ , with wrap-around on the indices.

This structure is enforced by averaging along diagonals in the following steps. At training epoch  $t$ , let

$$\begin{aligned}\hat{\mathbf{M}} &= \mathbf{W}_t \mathbf{W}_t^T \\ \Omega(\hat{M}(i, j)) &= \frac{1}{n} \sum_{k=1}^n \hat{M}((i+k)|_n, (j+k)|_n) \\ \mathbf{M}_t &= a\mathbf{M}_{t-1} + (1-a)\Omega(\hat{\mathbf{M}}_t)\end{aligned}\tag{4.2}$$

### 4.3.3 Index Map Featurisation

There are two index map featurisation modules which are orientation IMF and scale IMF.

#### 4.3.3.1 Orientational IMF

An orientation IMF module has three steps, which are folding, unification, and feature encoding. The general workflow is demonstrated in Fig. 4.2.

Folding was used in the popular Swim transformer(2021) [79] for concatenating nearby pixel values into a vector at the centre point. Suppose there is a featurisation window whose width is  $k \times k$ , a folding operator gathers information from neighbours and outputs  $k^2 \times c$  channels,  $c$  is the channel depth before folding. An example is demonstrated in Fig. 4.2.

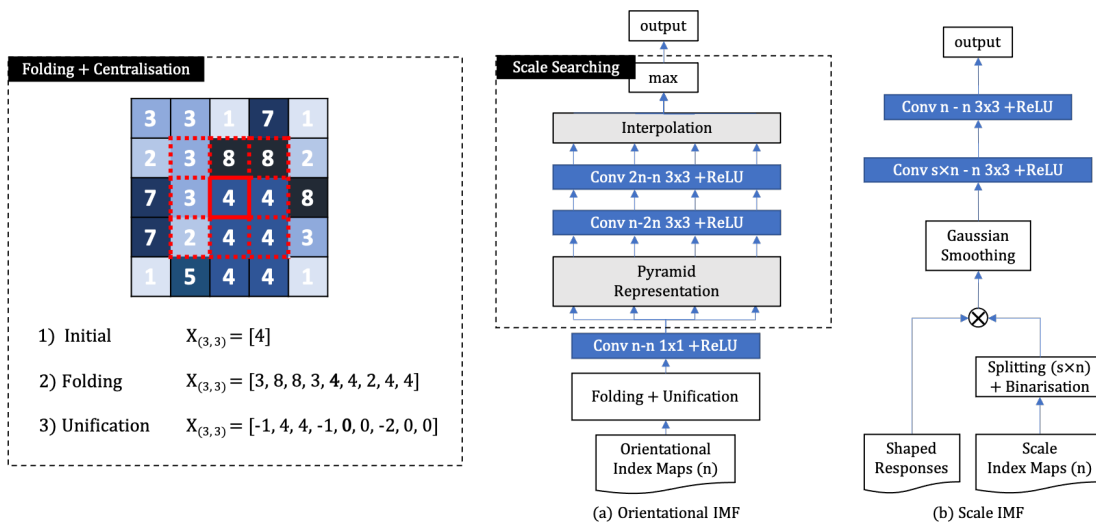


Figure 4.2: Left: A demonstration of the folding and centralisation process. Right(a): The processing flow of a typical orientational IMF module. Similar to MAC units, IMF kernels are also sliding on different scales to find useful patterns. Right(b): The demonstration of a scale IMF module, which takes a different encoding procedure. The index maps are firstly split into  $s \times n$  channels,  $s$  and  $n$  are the number of used scales and used parallel units respectively. Each split channel highlights the area of corresponding selected scale. The area is expanded by a Gaussian kernel and then weighted by the shaped response score at every position.

These collective information  $\mathbf{x}$  will be unified by removing the central index. Then the unified indexes will be analysed by the following layers.

In practice, the computation of orientational IMF at each pixel can be simplified as,

$$\begin{aligned}
 \text{IMF}(\mathbf{x}) &= b + \sum_{i=1}^{k^2} w_i (x_i - x_{\text{center}}) \\
 &= b + \sum_{i=1}^{k^2} w_i x_i - \sum_{i=1}^{k^2} w_i x_{\text{center}} \\
 &= (b + \sum_{i=1}^{k^2} w_i x_i) - x_{\text{center}} \sum_{i=1}^{k^2} w_i
 \end{aligned} \tag{4.3}$$

---

**Algorithm 1** Orientational Index Map Featurisation (python)

---

- 1: **function** ORIENTATIONAL IMF(X, w)
  - 2:     feature = Conv2D(weights=w, input=X)
  - 3:     bias = sum(w)\*X
  - 4:     result = feature-bias
  - 5: **return** result
- 

This is the basic case. Considering the scale of an object is same on index maps and feature maps, the orientational encoding should also be operated on pyramid representations. Only the maximum signal is returned, shown as plot (a) in Fig. 4.2.

#### 4.3.3.2 Scale IMF

The procedure of scale indexes featurisation is slightly different. As discussed in Section.4.2.4, the inputs will be split into  $s$  channels at first, which equals the number of scales used in pyramid pooling. Each channel is a binary representation in which the pixels responding at the indexed scale are activated. It is then multiplied with the response shaping results. Not just the scale information, but also the confidence about the information are encoded.

To prevent gradient vanishing, a small positive value is added to the maps, and the new maps will be smoothed by a  $5 \times 5$  Gaussian kernel. This is for enlarging the effective area of a scale choice to help overcome transition breaks or small obstruction gaps. Two convolutional layers with ReLU are deployed for feature extraction in the scale IMF module.

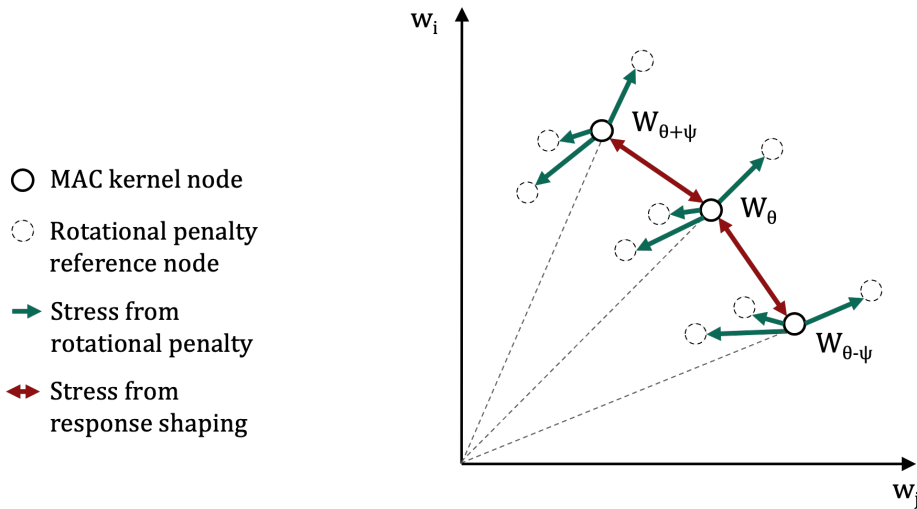


Figure 4.3: A simplified illustration of how a G-MASC model locates the directional kernel nodes in a  $k^2$  dimensional space. The distances between directional kernels are managed by the response shaping stress. For every individual kernel, its position is further depicted by a distance vectors between itself and a group of rotatable reference nodes that are projected to the corresponding positions in advance. The number of reference nodes is a hyper-parameter of G-MASC, which is related to the used kernel size. Stress is imposed on the row variance of a distance matrix which is a combination of all the distance vectors (columns).

#### 4.3.4 Rotational Penalty

Response shaping is built based on an expectation over kernel-wise distances. We have mentioned that underdetermination as an intrinsic problem of the original MASC. It also reflects the dilemma of balancing direction granularity and kernel size. Keeping directional kernels aligned, a larger kernel would require more constraints as there are more parameters to be updated.

The plan was to increase the number of directions which would give more kernel-referenced constraints (kernel-wise correlation), but this will also increase the number of base kernels due to the fixed re-use degree. Unlike steerable filters or circular basis which are analytical determined, the rotation symmetry of MASC is encouraged by the constraints used at each iteration. It is possible to bring regularisation from references. These reference nodes should have constant relative pair-distance to directional kernels. This is the principle of phase-offset rotational error (PoRE).

The goal of PoRE is to find the closest niche positions for a set of directional kernels. The fundamental strategy can be described as 'think globally and act locally',



a phrase used by local linear embedding (LLE) [102]. The position of each directional kernel in the kernel space is measured by  $n_{\text{ref}}$  reference nodes. Suppose there is a set of precisely rotated kernels  $\mathbf{w}$ . Though,  $\mathbf{w}$  can be any steerable function, its expression at a particular orientation should have a stable distance to another rotatable reference  $\mathbf{g}$  at the same orientation or with a fixed phase offset. For example,  $\mathbf{g}(\theta)$  is a reference function at  $\theta$  direction.

$$d_i = \frac{1}{k^2} \|\mathbf{w}_i - \mathbf{g}(4i\pi/n + \psi)\| \quad (4.4)$$

Here, with an example of 16 directions, distance  $\mathbf{d}$  is made by two parts, which are 8 directional kernel distances and 8 reflected kernel distances. In Eq.4.5,  $d_i$  represent the mean of directional kernels, and  $\bar{d}_r$  represents the mean of reflected directional kernels,  $\mathbf{G}$  is a collection of reference kernels  $\mathbf{g}_i$ .

To be more specific, pattern  $\mathbf{w}_i$  should have similar distance to the patterns  $\mathbf{g}(4i\pi/n + \psi)$  as  $\mathbf{w}_j$  to  $\mathbf{g}(4j\pi/n + \psi)$ . We can view it from the aspect of dimension compression. The process indeed is compress a  $k^2$  dimensional feature to the space of  $n_{\text{ref}}$ . Because the precise expectation of  $d_i$  can be directly measured, one can evaluate them by computing the variance of individual distances. A small variance means stronger consistency. Squared error is chosen as the distance metric here.

$$\sigma^2(\mathbf{W}, \mathbf{G}) = \frac{1}{n} \sum_{i=0}^{n/2-1} (d_i - \bar{d})^2 + \frac{1}{n} \sum_{i=n/2}^{n-1} (d_i - \bar{d}_r)^2 \quad (4.5)$$

Orientated ridge patterns derived from a Gabor function are used as the reference here. The parameter to optimise is  $\mathbf{w}$ , and standard deviation is employed as a measurement of the relative distance consistency. To make the combination of all reference nodes a full-rank matrix, instead of using different Gabor reference functions, filters with different orientation (phase offset) are compared with each directional kernel.

$$\text{PoRE}(\mathbf{W}) = \frac{1}{n_{\text{ref}}} \sum_{m=1}^{n_{\text{ref}}} \sigma^2(\mathbf{W}, \mathbf{G}_m) \quad (4.6)$$

There are more options for crafting a reference node. It can be constructed by a single Gabor function, or a combination of several Gabor functions as well, once they are full rank. Parameter  $\lambda$  is the loss coefficient, since the penalty will then be added to the semantic BCE between ground truth and model output as,

$$L = \text{BCE} + \lambda_{\text{PoRE}} \text{PoRE}(\mathbf{W}) \quad (4.7)$$

Here, the function  $BCE$  represents binary cross-entropy.

In practice, as the aim is to control intra-class balance, the PoRE is only applied on base kernels under the asymmetric assumption. However, if one chooses to use a symmetric assumption, both base kernels and the transposed kernels should be included. In this case, transposed kernels are regarded as the rotation equivalents.

### 4.3.5 Why use squared residual but not correlation?

Correlation is used in PoC as a rotation symmetry evaluation of a set of kernels. PoRE which is based on squared distance is more suitable as a cost function. The two main reasons are as follows:

The two metrics have different focuses. During training, directional weights keep changing when steerable reference nodes are randomly created and stay constant. Squared error is still sensitive even all the pairs are at a great distance. On the contrary, correlation compresses its change when distance goes large.

Square error is also commonly used in manifold embedding tasks. PoRE and PoC are based on a naive assumption which will make more sense on significant pair discrepancy but not on subtle distance inconsistency. This meets the mentioned characteristics of squared distance metric.

The second reason is to avoid gradient vanishing. When training a model with PoC in its cost function, significant weight inflation was observed on kernel weights before normalisation. In Eq.4.8,  $||\mathbf{W}||$  is large in the denominator. If calculating the partial derivatives of the equation on all  $w$ , we will see gradients approaching zero when having a large  $w$ . The weights can raise to as great as tens of thousands. It causes gradient vanishing and leads to stagnated updating.

$$\mathbf{W}_{\text{normalised}} = \frac{\mathbf{W}}{||\mathbf{W}||} \quad (4.8)$$

However, these issues can be mitigated by using PoRE. It offers a milder than linear stress when the variance is small and renders stronger updates when the condition is serious. Kernel weights are curbed before inflation occurs.

PoRE can be interpreted as a regularisation that squeezes all the weights in a set towards a particular number. Further evidence of this design will be discussed in the experiment section.

With PoRE, we can be more confident about a MASC model. Nevertheless, one should be careful in practice since strong rotational guidance can be neutralised and even

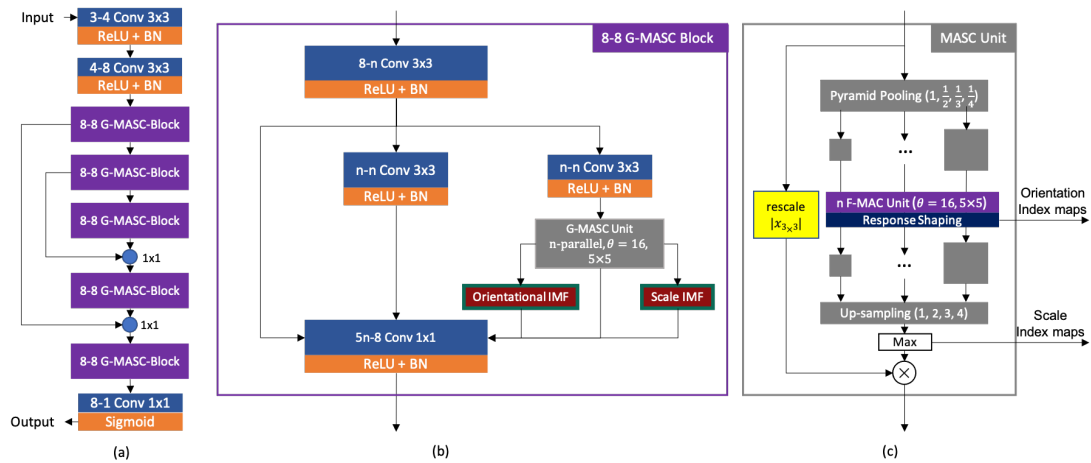


Figure 4.4: The visualisation of an G-MASC-5-2-16 example. The architecture is very similar to the original MASC's but with the additional orientational and scale IMF modules aside the MASC unit. The index maps are derived from the response shaping and the pyramid scale selection. This architecture is built specially for the axon, retina and cell nuclear segmentation tasks. For G-MASC application on the other images, more or less G-MASC blocks could be appropriate.

mislead the gradient from label comparison.

### 4.3.6 Model Structure

The structure of a G-MASC model is generally the same as the one used in the original MASC, which is a cascade structure with skip connections. Inside a G-MASC block, the new orientational and scale IMF modules follows MASC units, shown in Fig. 4.4(a). The width of all middle layers is set to the number of parallel units. Specifically, the pre G-MASC layers are also arranged with the number of channels to the equal to the number of groups, which means it processes features independently for each G-MASC unit. The last layer preparing a G-MASC block's output is also widened. The parameter number is increased compared to the original algorithm.

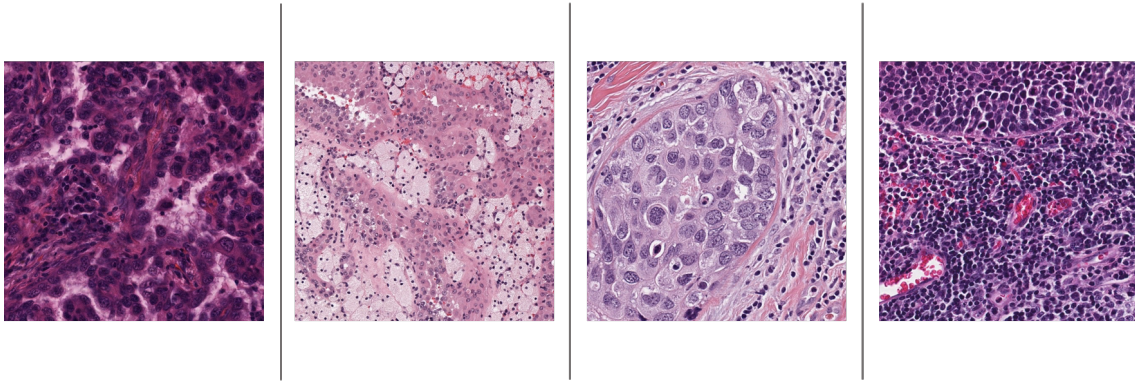


Figure 4.5: The training set of MoNuSeg dataset contains 30 carefully annotated tumor tissue images collected from multiple hospitals and institutions. These images covers different organs. The original size of them were  $1000 \times 1000$ , and cropped to 8,000 patches for data augmentation and memory optimisation consideration.

## 4.4 Experiment

We evaluated G-MASC on three datasets, CHASE-DB1, DRIVE, and MoNuSeg [66].

The first two were used in testing the original algorithm. The new MoNuSeg containing cell nuclear images was included in order to show the algorithm’s difference. These data are formatted as RGB images in  $1,000 \times 1,000$  resolution, collected from several patients’ different organs and who were diagnosed at multiple hospitals. The MoNuSeg dataset has been separated into training and testing sets in advance. Its training set contains 30 images of around 22,000 cell nuclear annotations, while the testing set has another 7,000 annotations in 14 images.

The model of G-MASC follows the same nomenclature of MASC. A G-MASC 5-2-16 5x5 refers to a model with 5 G-MASC blocks, each has 2 parallel directional kernel sets of 16 kernels (8 directions + 8 reflections). The all kernels have a size of  $5 \times 5$ . Note that the difference between models is not only about their G-MASC modules. For example, G-MASC 5-2-16 and G-MASC 5-6-16 have different junction channel numbers, and this can be realised from the total parameter changes in the table.

### 4.4.1 Environment

There are 8,000 patches randomly cropped for each dataset’s training set. In each test, original-sized images were fed to the models.

In the general performance experiment, models are trained with all the samples. In the ablation and hyper-parameter studies, to avoid performance saturation, only half of

the training set is used for showing more granularity from case to case.

All the presented figures are averaged from a 5-fold cross validation. There are 6,400 and 3,200 training samples used for the two conditions. We expect better performance to be achieved by either involving more examples during training or increasing the model scale.

The objective and the main characteristic of G-MASC is its modelling efficiency. In order to justify this point, our focus is on using the fewest parameters to get a competitive performance.

#### 4.4.2 General Performance

The AUC (area under the curve) metric scans through all valid thresholds and is commonly used for evaluating general performance. Computed the ROC for each G-MASC variant and tried to find those models with the fewest parameters which still achieved an AUC withing  $\pm 0.2$  of the best benchmark result.

The experiment results in Table.4.1 show that G-MASC can achieve performance similar to that of VesselNet but using only 0.9% of the parameters on CHASE-DB1, or 5.7% of the parameters on the DRIVE dataset. And on the cell nucleus task MoNuSeg, the advantages of MASC methods are even more obvious. Methods such as MiUNet [55] and IterMiUnet [65] uses dense connection and multiple iterations, which require higher computational density.

Comparing it with the original MASC models, on DRIVE dataset, we trained two variants, a compact G-MASC 5-2-16 5x5 and an extended G-MASC 5-6-16 11x11. The results show that if both G-MASC and the original MASC use a compact setup (5.4K and 3.8K parameters), the G-MASC model can outperform its MASC counterpart by 1.0% in sensitivity and 0.6% in AUC. On the cell nuclear segmentation task, G-MASC outperformed the MASC with 76.3% (73.6%) F1 score and 94.5% (93.6%) AUC.

These show G-MASC has advantages compared to the original MASC, especially when using fewer directions and smaller kernels. If both use an extended setup, the vessel task performances of them would saturate as shown in Table.4.1.

Regarding to the MoNoSeg test, the G-MASC model can provide more rotationally congruent patterns after training. This fact is also reflected in the metrics, where the G-MASC wins a percentage of 2.7% on F1 score, 2.6% on accuracy. Due to the over-segmentation effect which was confirmed from the output masks, the original MASC has a high sensitivity score.

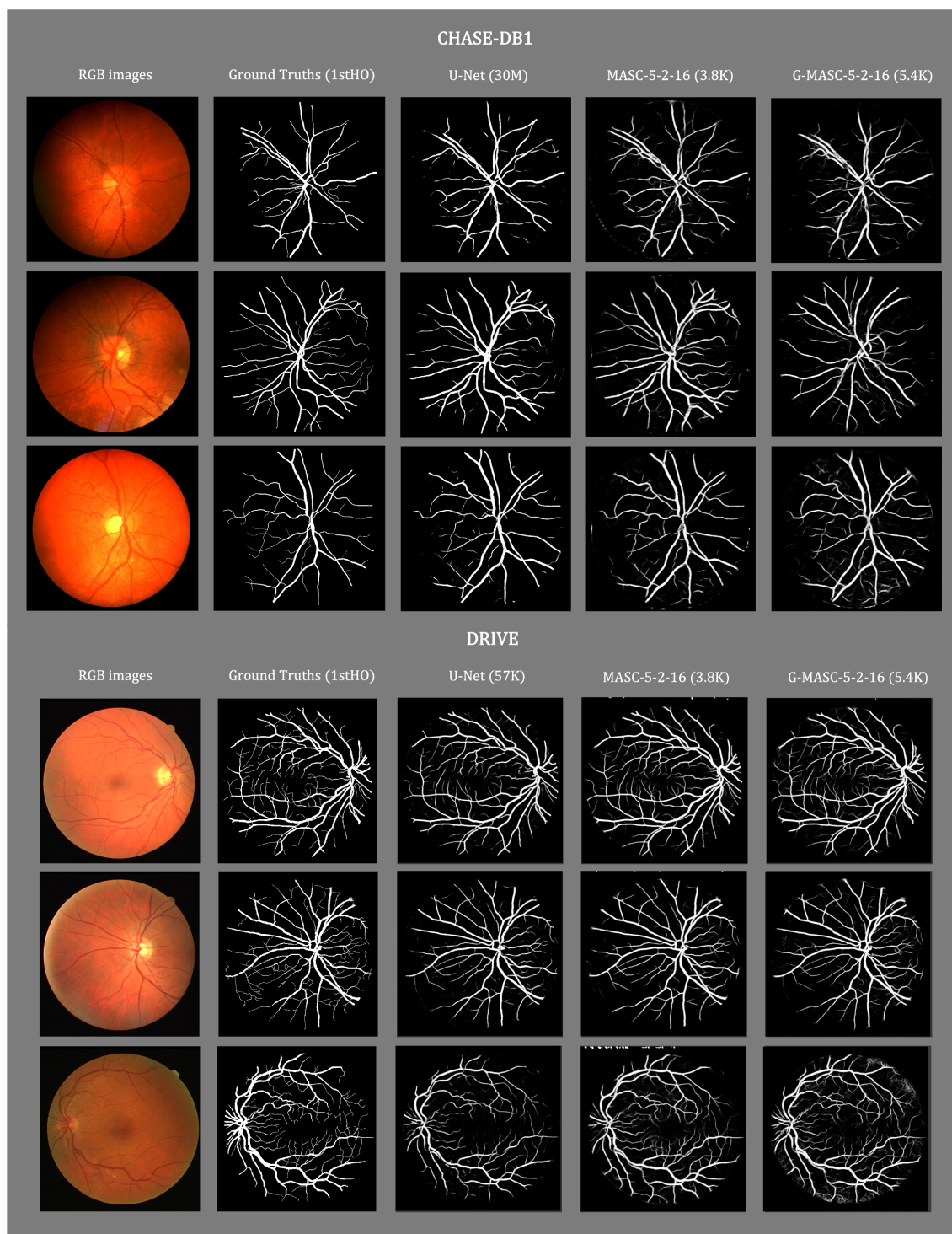


Figure 4.6: The segmentation results on CHASE-DB1 and DRIVE.



Table 4.1: Comparison with other methods

Datasets	CHASE- DB1	F1	Se	Sp	Acc	AUC	Para.
CHASE-DB1	2nd Observer	76.9	74.3	97.9	94.7	-	-
	U-Net	<b>79.2</b>	79.1	97.4	95.4	95.3	57.4K
	VesselNet [127]	79.1	78.2	<b>98.1</b>	<b>96.2</b>	97.6	585.8K
	MiUNet [55]	74.1	84.4	95.7	95.7	94.7	69K
	IterMiUnet [65]	78.8	<b>84.4</b>	97.0	95.9	<b>98.1</b>	150K
	MASC 5-2-16 5x5	80.2	80.7	97.8	95.6	97.4	<b>3.8K</b>
	G-MASC 5-2-16 5x5	79.4	76.4	98.0	95.6	97.4	5.4K
DRIVE	2nd Observer	78.8	77.6	97.3	94.7	-	-
	U-Net	81.4	75.4	98.2	95.3	97.6	30.6M
	VesselNet [127]	81.3	76.3	<b>98.4</b>	95.6	97.7	585.8K
	MiUNet [55]	82.4	79.7	98.0	<b>95.7</b>	<b>98.1</b>	69K
	MASC 5-2-16 5x5	79.1	77.1	97.4	94.7	96.4	<b>3.8K</b>
	G-MASC 5-2-16 5x5	79.6	78.1	97.7	95.1	97.0	5.4K
	MASC 5-6-16 11x11	81.6	<b>80.6</b>	97.4	95.4	97.5	19.6K
G-MASC 5-6-16 11x11	81.0	78.0	97.9	95.5	97.5	33.6K	
MoNuSeg	U-Net(Double Conv)	71.4	72.5	90.1	86.7	91.4	57.4K
	MASC5-2-16 7x7	73.6	<b>86.3</b>	85.8	85.9	93.6	<b>4.3K</b>
	G-MASC5-2-16 7x7	<b>76.3</b>	81.4	<b>90.5</b>	<b>88.5</b>	<b>94.5</b>	5.9K

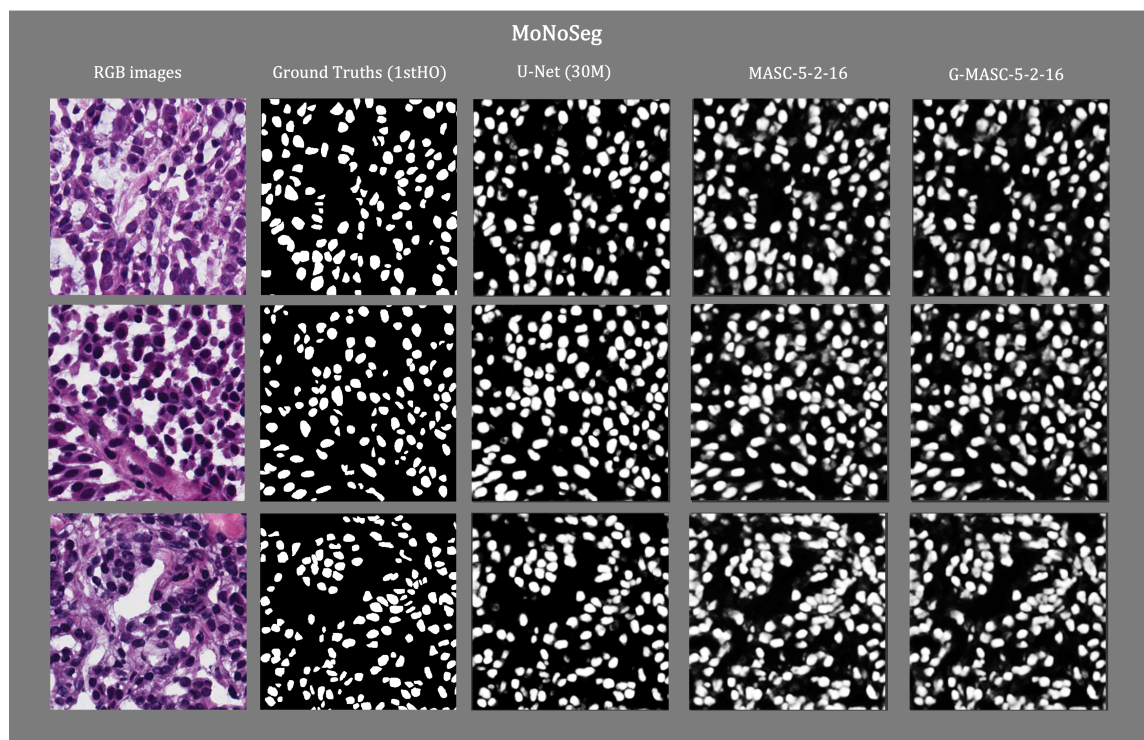


Figure 4.7: The segmentation results on MoNuSeg.

Table 4.2: Ablation Study on DRIVE(4,000 samples for 5-fold cross validation), Scale-IMF refers to scale index map featurisation module, Orientation-IMF refers to orientation index map featurisation module.

Methods	F1	Se	Sp	Acc	AUC	G
G-MASC-5-2-16 $5 \times 5$	<b>79.4±0.1</b>	<b>76.2±0.7</b>	97.7±0.1	<b>95.0±0.1</b>	<b>96.9±0.1</b>	<b>86.3±0.3</b>
w/o IMF, $\lambda_{\text{PoRE}} = 0$	74.8±1.2	66.9±2.8	<b>98.3±0.4</b>	94.3±0.3	95.9±0.3	81.0±1.5
$\lambda_{\text{PoRE}} = 0$	79.1±0.2	75.6±0.7	97.7±0.2	94.9±0.1	96.7±0.1	86.0±0.4
w/o IMF	79.1±0.5	76.1±0.6	97.6±0.3	94.9±0.2	96.7±0.1	86.2±0.3
w/o Scale-IMF	79.3±0.2	76.1±0.5	97.7±0.1	95.0±0.0	96.8±0.3	86.2±0.3
w/o Orientation-IMF	79.4±0.2	75.6±0.6	97.8±0.1	95.0±0.1	96.8±0.1	86.0±0.3

Compared to CHASE-DB1, DRIVE has a biased vessel centre and many tiny branches. We trained a model without PoRE on CHASE-DB1, and found the result is very close to the model with PoRE, and the PoC is steady as well. Again, this indicates that for a type of datasets on which a model can achieve a good set of rotated kernels with the simpler loss function, the benefit from PoRE may not be significant.

#### 4.4.3 Ablation Study

Ablation experiments were performed to explore the contribution of each module. One objective of updating G-MASC from the original curvilinear MASC was to provide stronger and more controllable rotation symmetry supervision. This may not make a difference on a dataset such as CHASE-DB1 in which the orientated vessels are spread with almost equal frequency in all directions and relatively easier to keep a steady update.

The experiment was conducted on a module basis, including PoRE and the two index map featurisation modules. In Table.4.2, the rotational penalty is switched off by making  $\lambda_{\text{PoRE}} = 0$ . All the models have the same configuration of G-MASC-5-2-16  $5 \times 5$ .

Comparing the G-MASC model and the original alike model(w/o IMF,  $\lambda_{\text{PoRE}} = 0$ ), the additional modules are found to bring significant improvement as 5% F1 score and 8.7% sensitivity. Also, the results of G-MASC models have a much smaller range. This suggests that with the same backbone structure, a new G-MASC block can outperform a MASC block in a more stable manner. On the other hand, the G-MASC model has a better balance between the classes since its G score (geometric mean of sensitivity and specificity) equals to 86.4% when the number for the original MASC is 81.0%. If looking into the sensitivity(Se) and specificity(Sp) columns, G-MASC is stronger in



finding the true positives than the opposite.

The figures for the model without any of PoRE, orientation-IMF, and scale-IMF are close on most metrics. This may be due to the saturation of modelling capacity with 4,000 samples. It was found in the hyperparameters studies that the marginal increment attenuates when the general AUC is approaching 97% as well. Besides that, as either response maps (the output of directional kernels after response shaping, affected by PoRE) or the IMF outputs are derived from the same sources, the corresponding G-MASC kernels, this information may lead to a stronger but correlated conclusion. A change in the scale selection can indicate a change in the orientation selection, they are linked. If a response map has been sufficient to recognise the desired classes whatever an object's scale or orientation is, then these additional clues may not be that influential. Suppose that there is a task about predicting the vessel orientation at each position, then the orientation-IMF is important for a model.

#### 4.4.4 PoRE Rotational Cost

The PoRE plays an essential role in ensuring a convolutional model has precise and accurate orientational detection. The entire MASC system is built on this objective. It was not guaranteed and under a loose control due to the ' $n < d$ ' problem in response shaping until being regularised by PoRE. Though it is not necessary to be an obtrusive factor in all cases, PoRE is important for the tasks where keeping rotation symmetry is hard due to many reasons. Here we will explore how PoRE works. It is important to note that the relationship between rotation symmetry and modelling accuracy is more complex than intuition, and is not monotonic.

We have explained the objective of PoRE is searching for a set of orientation-aligned positions in the kernel space that is closest to the positions guided by the task labels. It is clear that the two guidances can have a disagreement, even very opposite in some rare cases. In the other cases, rotation symmetry can be well maintained by response shaping. For example in CHASE-DB1, Fig. 4.6, the vessels are centred at the middle of images. It has been shown in Chapter.3 that the system can deal with this problem. PoRE further optimised the modelling by tightening the range of results.

During development, we designed two parameters to control PoRE, the number of reference nodes  $n_{\text{ref}}$  and the function weightage  $\lambda_{\text{PoRE}}$ . Both have been defined in Eq.4.7. They influence different aspects of the function. The  $\lambda_{\text{PoRE}}$  manages the step length, revealing how the PoRE gradient is associated with the semantic gradient. Here,  $n_{\text{ref}}$  is related to the kernel function, we are curious about whether it is necessary

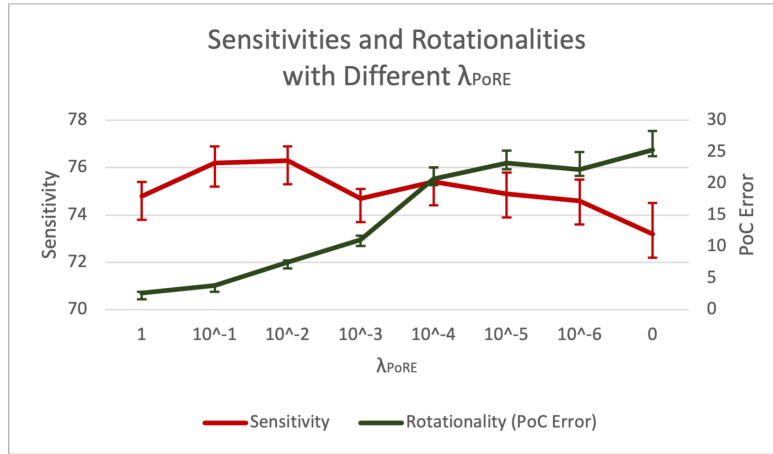


Figure 4.8: Sensitivity score drops when having a poor rotation symmetry performance (greater PoC error).

to give clear rotation symmetry guidance which may squeeze the reachable area in the kernel space. Again, PoRE does not promise rotation symmetry, however, it makes good rotation symmetry more likely.

The rotational error level will be heading to 25.3 when none additional constrain is applied and kernels solely relying on response shaping. If looking into Table.4.4, most peaks concentrate on the rows using  $\lambda_{\text{PoRE}} \in [0.1, 0.01]$ . Also, in Figure. 4.8, we could see that the sensitivity goes down when the PoC error rises. As suggest above, a very precise rotational expression does not necessarily result in good detection. When  $\lambda_{\text{PoRE}} = 1$ , PoC reaches its lowest level in the table, but the sensitivity dropped. Forcing the directional kernels is possible to divert the kernel away from its optimal evolution trajectory. On the other side, setting  $\lambda_{\text{PoRE}} = 0$  would raising the standard error of metrics.

The constraints on directional kernels should be neither overdetermined nor under-determined, this means accordingly increasing the number of references when using a larger kernel. However, in practice, the relationship between reference number and kernel scale is not straightforward, as the solution requires an iterative-based method rather than analytical.

In Table.4.4, comparing ( $k = 5 \times 5$ ,  $n_{\text{ref}} = 25$ ) with ( $k = 5 \times 5$ ,  $n_{\text{ref}} = 33$ ), if considering the effect from calibrated response shaping, both cases are overdetermined. The PoC of  $n_{\text{ref}} = 33$  is significantly lower than the  $n_{\text{ref}} = 25$ , which confirms the hypothesis of using more references will further regularising the directional kernels. The sensitivity of  $n_{\text{ref}} = 33$  is 1.1% higher.

Table 4.3: Cost Function Weights Study on DRIVE Dataset (4,000 samples for 5-fold cross validation)

Weights	Se	Sp	G	F1	PoC
$\lambda_{\text{PoRE}} = 1$	74.8±0.6	97.9±0.1	85.6±0.3	79.0±0.3	<b>2.6±0.2</b>
$\lambda_{\text{PoRE}} = 10^{-1}$	76.2±0.7	97.7±0.1	86.3±0.3	<b>79.4±0.1</b>	3.8±0.1
$\lambda_{\text{PoRE}} = 10^{-2}$	<b>76.3±0.6</b>	97.7±0.1	<b>86.3±0.3</b>	79.3±0.3	7.5±0.3
$\lambda_{\text{PoRE}} = 10^{-3}$	74.7±0.4	<b>97.9±0.1</b>	85.5±0.2	78.9±0.3	11.1±0.6
$\lambda_{\text{PoRE}} = 10^{-4}$	75.4±0.6	97.8±0.1	85.9±0.3	79.2±0.3	20.7±1.8
$\lambda_{\text{PoRE}} = 10^{-5}$	74.9±0.9	97.9±0.2	85.6±0.4	79.2±0.2	23.2±2.0
$\lambda_{\text{PoRE}} = 10^{-6}$	74.6±0.9	97.8±0.1	85.4±0.5	78.5±0.5	22.2±2.8
$\lambda_{\text{PoRE}} = 0$	73.2±1.3	98.0±0.2	84.7±0.6	78.2±0.3	25.3±3.0

Comparing ( $k = 5 \times 5$ ,  $n_{\text{ref}} = 33$ ) with ( $k = 7 \times 7$ ,  $n_{\text{ref}} = 33$ ), the latter is well defined as its  $n_{\text{ref}} = 33 = 49 - 16 = k^2 - n$ . With the same number of reference kernels, it is found using a larger kernel is beneficial to the general performance.

Comparing ( $k = 7 \times 7$ ,  $n_{\text{ref}} = 33$ ) with ( $k = 7 \times 7$ ,  $n_{\text{ref}} = 49$ ), then we could see that the case using  $k^2 - n$  references outperforms the case using  $k^2$  references by 0.4% AUC. This confirms the previous conclusion that the response shaping factor should be included when gauging regularisation degree.

Comparing ( $k = 7 \times 7$ ,  $n_{\text{ref}} = 49$ ) with ( $k = 9 \times 9$ ,  $n_{\text{ref}} = 49$ ), we saw the latter one has a greater 0.4% AUC. The results shows that even a model that has larger but underdetermined kernels can still beat a model with the same number of reference but smaller kernels size. This re-confirmed the conclusion from the first comparison pair.

The ( $k = 9 \times 9$ ,  $n_{\text{ref}} = 49$ ) and ( $k = 9 \times 9$ ,  $n_{\text{ref}} = 81$ ) cases can be regarded as  $k^2 - n \pm a$ . It indicates that, with a similar value  $a$ , the case of  $k^2 - n - a$  acquired 2% sensitivity greater than the case of  $k^2 - n + a$ . In the other words, if one has to make a choice between overdetermination and underdetermination and the severity is approximately the same, an underdetermined model could be a more confident choice.

To summarise this experiment, given a G-MASC model,

1. Performance can be improved by using a larger kernel size;
2. From the current information, the recommendation is using  $k^2 - n$  reference nodes;
3. The performance drops slower along the underdetermination side than the overdetermination side.

Table 4.4: Kernel Size and Number of References Study on DRIVE Dataset (4,000 samples for 5-fold cross validation)

Kernel Sizes	F1	Se	Sp	Acc	AUC	G	PoC
$k = 5 \times 5, n_{\text{ref}} = 25$	79.5±0.1	75.1±0.4	98.0±0.1	95.1±0.1	96.9±0.1	85.8±0.2	4.6±0.1
$k = 5 \times 5, n_{\text{ref}} = 33$	79.4±0.1	76.2±0.7	97.7±0.1	95.0±0.1	96.9±0.1	86.3±0.3	<b>3.8±0.1</b>
$k = 7 \times 7, n_{\text{ref}} = 33$	79.6±0.3	75.6±0.2	97.9±0.1	95.1±0.1	97.0±0.1	86.0±0.1	5.4±0.3
$k = 7 \times 7, n_{\text{ref}} = 49$	78.8±0.2	75.0±0.7	97.8±0.1	94.9±0.1	96.6±0.1	85.6±0.4	5.3±0.5
$k = 9 \times 9, n_{\text{ref}} = 49$	<b>80.0±0.3</b>	<b>76.9±0.3</b>	97.8±0.1	95.1±0.1	97.0±0.1	<b>86.7±0.1</b>	4.5±0.1
$k = 9 \times 9, n_{\text{ref}} = 81$	79.6±0.2	74.9±1.0	<b>98.1±0.2</b>	<b>95.2±0.0</b>	<b>97.1±0.1</b>	85.7±0.5	4.0±0.2

Table 4.5: PoC vs. PoRE on DRIVE Dataset(4,000 samples for 5-fold cross validation)

Methods	F1	Se	Sp	Acc	AUC	G	PoC
$\lambda_{\text{PoRE}} = 0$	78.2±0.3	73.2±1.3	98.0±0.2	94.8±0.1	96.7±0.1	84.7±0.6	25.3±3.0
PoC	78.3±0.4	74.5±1.0	97.7±0.3	94.8±0.2	96.5±0.2	85.3±0.4	<b>2.7±0.3</b>
PoRE	<b>79.4±0.1</b>	<b>76.2±0.7</b>	<b>97.7±0.1</b>	<b>95.0±0.1</b>	<b>96.9±0.1</b>	<b>86.3±0.3</b>	3.8±0.1

In chapter 3, a larger kernel size was shown to improve performance but made it harder to maintain consistent rotation kernels. In G-MASC, PoRE as a solution needs to be given a set of  $n_{\text{ref}}$  reference functions (such as Gabor). The difference between the performance with small and large kernels was not as large for our approach. A smaller kernel is an economic choice. The general PoC level is stable, this indicates that the rotation for larger kernels is well controlled.  $\lambda_{\text{PoRE}}$  is the penalty weights. The  $\lambda_{\text{PoRE}}$  experiment shows that both too small or large values of PoRE are detrimental to the model’s ability to detect true inputs (see the Se columns).

#### 4.4.5 Compare PoRE with PoC

In the motivation section, we discussed the reason for choosing PoRE instead of PoC as rotation symmetry supervision. To further justify this point, we performed more experiments. In Table. 4.5, the model with PoRE is found able to outperforms both PoC and the non-supervised version. Due to the gradient vanishing problem, the directional kernels in a G-MASC model under PoC are not effectively trained and this could be worse than having no rotation symmetry supervision at all.

#### 4.4.6 Update M in Different Ways

Momentum was proposed in the original MASC for stabilising **M**. However a smoother change sometimes is not sufficient to cope with complex images, and thus we designed the calibrated response shaping in G-MASC to make it also unified.

Table 4.6: Momentum Study on DRIVE Dataset(4,000 samples for 5-fold cross validation)

Methods	F1	Se	Sp	Acc	AUC	G	PoC
momentum=0.99	<b>79.4±0.1</b>	<b>76.2±0.7</b>	<b>97.7±0.1</b>	<b>95.0±0.1</b>	<b>96.9±0.1</b>	<b>86.3±0.3</b>	3.8±0.1
momentum=0	78.3±0.4	74.6±1.3	97.7±0.3	94.7±0.2	96.7±0.2	85.3±0.7	<b>3.8±0.3</b>

Comparing Table.4.6 with Table.3.8 in Chapter 3, momentum plays a more critical role, not only obviously bringing the metrics' standard error down, but also boosting the model's F1 score and sensitivity by 1.1% and 1.6% respectively, when not hurting the rotation symmetry (PoC). Regarding the concern that too large a weight may push directional kernels to alignment at the cost of semantical analytic capacity, a slower update on  $\mathbf{M}$  acts as a form of regularisation, smoothing the overall trajectory of its values.

#### 4.4.7 Kernel Construction

The kernel construction method decides how it responds to gradients. One may challenge whether it is still needed to use a hybrid structure (Hadamard product between a weight matrix and a Gabor matrix), as G-MASC has had many protection mechanisms.

The same experiment of Table. 3.9 is done with G-MASC so a comparison can be made. In Chapter 3, we see that cancelling the Gabor kernel does not make a substantial change for the averaged ultimate score except bringing down the standard errors.

It is slightly different for G-MASC as PoRE is involved, the numbers in Table 4.7 suggest that the hybrid kernel scheme is still essential. Especially we saw cancelling it is at the cost of dropping accuracy from 96.7% to 83.6%, and F1 score from 79.4% to 57.8%. The Conv-only model gets a higher sensitivity on the cost of massive false positives due to a strong positive bias. On the other side, the best AUC (area under the curve) of the model can reach 94.4% which is 2.8% higher than the average number. The conv-only kernel is easier to train (the loss score changes faster at the beginning of training), and also more vulnerable to gradients. As a conclusion, the performance becomes weaker and more volatile when applying PoRE with  $\lambda_{\text{PoRE}} = 0.1$  on the conv-only directional kernels.

Table 4.7: Kernel Initialisation Study on DRIVE Dataset(4,000 samples for 5-fold cross validation), hybrid scheme and conv-only scheme

Methods	F1	Se	Sp	Acc	AUC	G	PoC
$W_{\text{hybrid}}, \lambda_{\text{PoRE}} = 0.1$	<b>79.4±0.1</b>	76.2±0.7	<b>97.7±0.1</b>	<b>95.0±0.1</b>	<b>96.9±0.1</b>	<b>86.3±0.3</b>	<b>3.8±0.1</b>
$W_{\text{Conv}}, \lambda_{\text{PoRE}} = 0.1$	57.8±3.9	<b>82.3±2.7</b>	83.7±4.0	83.6±3.3	91.6±0.9	82.7±1.7	7.2±0.8

Table 4.8: Kernel Initialisation Study on CHASE-DB1 Dataset(4,000 samples for 5-fold cross validation)

Methods	F1	Se	Sp	Acc	AUC	G	PoC
G-MASC $7 \times 7$	77.8±0.4	<b>75.9±0.8</b>	97.6±0.1	95.2±0.0	97.0±0.1	84.7±0.4	<b>18.9±0.5</b>
G-MASC $7 \times 7$ (cosine, $\gamma = 3$ )	78.6±0.2	74.8±0.9	98.1±0.1	95.5±0.0	97.2±0.1	<b>85.7±0.5</b>	10.1±0.6
G-MASC $7 \times 7$ (cosine, $\gamma = 12$ )	<b>78.6±0.3</b>	74.6±1.1	<b>98.1±0.1</b>	<b>95.5±0.0</b>	<b>97.3±0.0</b>	85.5±0.6	9.3±0.2

#### 4.4.8 Kernel Initialisation

The initialisation is a decisive factor for G-MASC models, just with the other hybrid models [81, 85]. There are infinity number of direction-aligned positions in the kernel space, it will make training easier for a G-MASC model to start from some sensible weights.

It was mentioned in Chapter.3 that some parameters of the Gabor kernel have a more significant impact, especially the variance  $\sigma$  and the aspect ratio  $\gamma$ . The two control the ridge width in horizontal and vertical directions respectively. It was found that keeping the ridge thin and long at least at the beginning can exaggerate the response discrepancy among directions and further strengthen the precision of response shaping. Another factor that is significant is the phase offset  $\phi_G$ . As basic textures, ridge ( $\phi_G = 0$ ) and edge ( $\phi_G = \frac{\pi}{2}$ ) are commonly found in images' principal components as a convolutional neural network's low-level feature detectors. We test a cosine initialised  $\mathbf{G}$  ( $\phi = \frac{\pi}{2}$ ). The results are presented in Table.4.8.

The results from cosine models are slightly better than a standard G-MASC. When we plot the exact directional kernels, it is found that the pattern learnt by a cosine model is different from what we used to see.

The kernel patterns are not constrained in its initial form but updated to a miscellaneous form. However, like what we learnt from the sine functions. the cosined G kernels tend to converge at a specific kind of pattern. Another difference between a sine and cosine initialised G-MASC is the latter has a higher chance trapped at a circular pattern position (isotropic) in the kernel space. In the case, the modelling capacity of a  $n$  directional G-MASC kernel collapses to one kernel, though computation is running on  $n$  bases. It undermines the efficiency in general.

## 4.5 Conclusion

The original MASC assumes target patterns are curvilinear. This assumption can strengthen the detection of axon-like objects by narrowing the searching area in parameter space. It also brings problems for generalisation. We introduce a model called G-MASC which explicitly creates a set of rotated and reflected kernels that can be used to estimate the orientation and scale of local structures. More explicitly speaking, the G-MASC tries to learn better large kernels with the same number of directional responses. The approach requires the estimation of far fewer parameters than other methods (such as U-Net) and can work on more common datasets. As a new version, G-MASC resolved the training problem of the original model, and can transfer index maps to additional features with new IMF models. It explores the potential of constructing comparable feature categories (orientations and scales).

For the axon tracing in Chapter 5, a MASC model is applied for generating object masks as its input data for efficiency consideration.

## Chapter 5

# Tracking Axon: Model-1

The methods in the previous chapter enable us to segment the individual objects on each frame. We now focus on tracking objects in order to understand how they change from one frame to the next.

Using a single model to complete both the tasks is easier to implement. However, since there is no available instance-level annotation, we have to develop a new approach.

We attempted to produce the labels for tracking manually, but soon found that it was not just time-consuming but also difficult as different axons can touch or cross, leading to ambiguities. The annotation quality cannot be guaranteed with limited human resources. It was realised that supervised tracking, at least at the beginning, is not an ideal choice in this situation.

The alternative plan we used, shown in Fig. 5.2, has two stages.

In the first stage, a heat-map-based model-1 is implemented to collect the information for discriminating axon individuals. Its outputs were then cleaned under both metric and human supervision. Human correction ensured that the tracking quality is precise enough to answer the biological questions. The resulting instance-level axon profiles will have two uses, to conduct analysis on them directly, and to wrap them as a new dataset for improving the accuracy of the algorithm.

In the second stage, a supervised model-2 was trained, which took the place of the model-1 in the system once its performance was confirmed. Many techniques could be used for this such as Graph Neural Networks(GNN) and instance segmentation. The first one can determine the target object like the model-1 but learns from its topological structure. The second would also take the place of the segmentation model.

The major challenges of axon tracking are illustrated in Fig.5.1. It is often observed



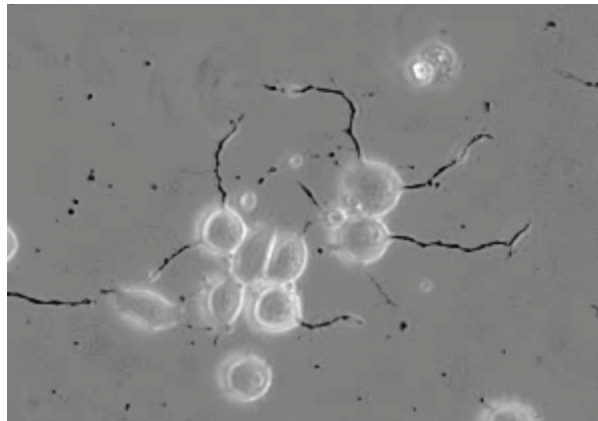


Figure 5.1: The plot demonstrates the main challenges of locating axon objects on an image. Entangled branches need to be separated, and breaks caused by overlapping or over-illumination will reconnect.

that axons can entangle with each other, and a good model should be able to assign branches to the right axon object. Also, axon segmentation can be influenced by many image issues, including overlapping and over-illumination.

In this chapter, we will focus mainly on the first stage. This includes the data collecting and cleaning plan, as well as the model building. We treat axons as tree(graph) structures. Tracking or separating axons is dealt with as a node classification task. The model will make the decisions for each node based on two questions; whether it is a part of a real axon, and whether it is a part of the current target axon.

## 5.1 General Plan

It is clear that manual correction inevitably involves subjective decision-making and biases the analysis to some extent. The robustness of the model-1 should still be able to guarantee the accuracy of biological conclusions, although it has to be manually corrected in the end. The solutions below require minimal correction and work is done in seven steps. The tracing model-1 is designed to process one axon object each run.

1. Build primary feature profiles for all positive-marked segments.
2. Clean up primary feature profiles and register them as axon objects which are formatted in tree data structures.
3. Determine a target object with the initial frame index and position, and start tracking.

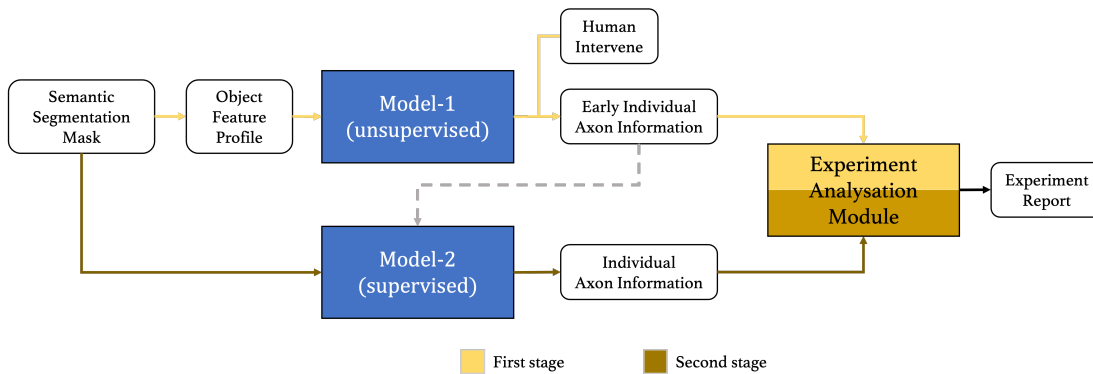


Figure 5.2: The implementation plan of this system. Different colours represent the procedures from the different stages. In stage-1 (light yellow), a heat-map-based decision model will be used to track and repair active axon objects. Individual-level annotations are generated at the same time. This crafted-feature algorithm plays a role as an annotation collector in the system. Together with manual correction, its prediction quality (dash line) is improved, then sent to train a supervised (deep yellow) model-2.

4. Search candidate objects (breaks) around the target, and put them into a candidate pool.
5. Refine the axon structure by executing prune (remove branches) and repair (merge with the candidates) alternatively for three rounds.
6. Move to the next frame. Pick the most matched axon object as the succeeding target object, then repeat steps 4 and 5 until meeting any of the three stop-tracking conditions.
7. Generate activity report which also includes a video log for human correction and analysis.

Of the steps, feature extraction is the most critical part of this model. Several rounds of cleaning and updating will be executed to improve the quality of the output. The general workflow is shown in Fig. 5.7.

After the main steps, additional robustness-boosting procedures are applied, including bi-directional tracking and human correction. Bi-directional tracking means applying the algorithm on both forward and reversed temporal directions. This allows the model to determine when a tracking error was emerging, the frame that a major tracking discrimination is detected. We have used this model to process 127 microscopy videos. In most scenarios, the model-generated results are satisfying, though

problem is inevitable. As is often the case with rule-based models when parameters are discovered from empirical searching, which can work poorly on unseen cases. In order to improve the utility of model-1, an additional manual supervision module was included for finding and fixing significant error, Such correction only needed to be applied on one frame, and the model can remember the correction (in an object activity history) and update its later decisions.

The tracking records were saved automatically every five frames. Thus a manual correction order will not re-run the entire task but from the latest checking point. Each piece of tracking data came with a visualised tracking log file to simplify the supervision work.

## 5.2 Feature Extraction

The axon information carried by the segmentation map needed further cleaning and distillation. At this stage, an isolated axon piece on these maps may not be a complete axon object for three reasons, 1) It can be a piece of debris or other visually similar substances but not an axon; 2) It does not contain all the branches of an axon which is broken due to segmentation error and obstacles; 3) Multiple axon pieces are sticking together. A feature extraction module in Fig. 5.7 will collect the required information and build a primary profile for every isolated piece as well as the cell objects. The following steps will focus on fixing these problems.

The segmentation maps returned by the network have continuous values (in the range [0,1]). We applied a threshold to obtain a binary response. Instead of using 0.5, an optimal threshold  $t$  is calculated from AUC analysis. It balances the classification performance on both positive and negative cases by maximising the geometric mean of sensitivity and specificity,

$$\begin{aligned} AUC(Se, Sp, t) &= \max_t(\sqrt{Se(t) \times Sp(t)}) \\ &= \max_t\left(\sqrt{\left(\frac{TP}{TP+FN}\right)\left(\frac{TN}{FP+TN}\right)}\right) \end{aligned} \quad (5.1)$$

The same binarisation method is applied to both axon and cell maps. Each isolated positive piece on these maps is counted as a candidate object and labeled with a unique index. Axon area skeletonization [139] is applied to get single-pixel-width traces.

We noticed that small blobs frequently appeared on axon traces as obstructions, for instance see Fig. 5.3. These objects may be of interest in the future, thus separate

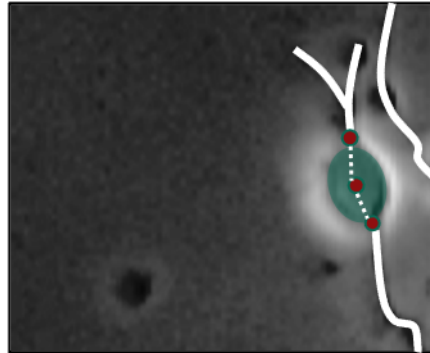


Figure 5.3: Axons and blobs are labelled separately. We fix a blob gap on an axon segment (solid lines) by connecting the gap tips to the centre point of the detected blob area (green ellipse) with straight estimations (dash lines).

segmentation maps for them are created. The blob gap on an axon is fixed by connecting two break tips to the middle point of the blob. As the blobs are normally small, a straight line is used.

### 5.2.1 First Round Cleaning

The first round of cleaning is based on the semantic relationship between cells and axons, and it is conducted under three policies.

Cells are a critical reference for axon tracking. Sometimes, a blob is identified as a cell by mistake (Fig.5.4 black arrow). The first policy requires removing any cell candidates that are smaller than a realistic size. This is quite straightforward, cell soma should have a minimal size, and false pieces are typically much smaller.

Though very rare, it is possible that a pixel is classified being both axon and cell. This is caused by a thresholding conflict. To resolve this issue, the second policy clarifies the priority of cell labels to be always higher than axon labels. Visually speaking, an axon normally has a short part rooted in the cell soma in the received microscope materials, Since this part of an axon is usually over-illuminated in the image, the standard measurement skipped it and counts from the first pixel out of the cell edge.

Every pixel is assigned with a semantic label, to be axon, cell, or background.

$$\text{Priority: Cell} > \text{Axon} > \text{Background} \quad (5.2)$$

An axon is a structure that grows out of a cell. Therefore, graphically, it must touch the boundary of any cell, and this should be expressed in the profiles. The second policy removes any candidates not attaching to a cell (Fig.5.4 red arrow). Note that an

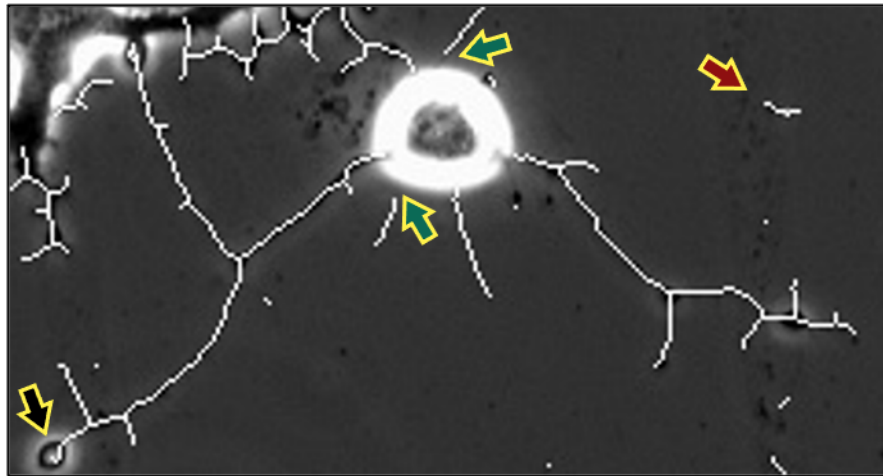


Figure 5.4: Some segmentation issues to be resolved by the first data cleaning. The black arrow shows that blobs may misguide the system by taking a slightly curvy route. Green arrows point the partition issue where axons may visually disconnected with neuron cells. This needs to be fixed in the tracking algorithm. The Red arrow indicates the false positive objects which have to be distinguished from real axon pieces in the repair step. These challenges have been solved by model-1 by the iterative connect and cut.

image border is registered as a special cell object to include the cases of axons growing from the outside of an image. This helps us to make cross-image measurements. Also, a binary mask may fail to reflect some delicate connections. A distance allowance is given to axons. This is achieved by applying morphological operations by dilation and erosion (Fig.5.4 green arrow) to bridge the small gaps between some cell bodies and their axons.

Most serious errors can be resolved by these three policies. After the first round of cleaning, we found the axon/cell data is ready for building the primary profile.

### 5.2.2 Building the Primary Axon Profile

The primary profile is in the form of information table, different from the later axon profile which is encoded in a tree structure. An example is given in Fig. 5.5, which contains the following attributes,

1. **Index:** Index for axon candidates is given on frame bases, and there is no guaranteed relation between the candidates with the same indexes, e.g. the No.7 record in Frame-1 is not necessary to be the same object represented by the No.7 in Frame-2.

label	connected_cells	uni_root	all_roots	growth_cones	box_left	box_top	box_width	box_height	num_nodes	total_length	longest
8	[2.]	[ 9 332]	[[ 9 332]]	[ 12 325]	325	9	8	4	8	8.24264	8.24264
9	[1.]	[11 84]	[[11 84]]	[12 78]	78	11	7	2	7	6.41421	6.41421
10	[3.]	[ 15 536]	[[ 15 536] ... [ 10 536]]		536	10	1	8	8	7	5
11	[2.]	[ 13 437]	[[ 13 437]]	[ 21 440]	437	13	4	9	9	9.24264	9.24264
12	[3.]	[ 12 493]	[[ 12 493] ... [ 13 492]]		492	12	2	2	2	1.41421	1.41421
14	[2.]	[ 17 405]	[[ 17 405] ... [ 27 400]]		400	17	11	11	23	24.4853	13.2426
16	[2.]	[ 19 417]	[[ 19 417]]	[ 20 422]	417	19	6	4	8	8.24264	8.24264
17	[2.]	[ 18 427]	[[ 18 427] ... [ 18 428]]		427	18	2	1	2	1	1
18	[3.]	[ 25 538]	[[ 25 538] ... [ 33 535]]		535	19	7	15	22	22.2426	15.8284
19	[2.]	[ 20 345]	[[ 20 345]]	[ 32 329]	329	20	17	13	22	23.8995	23.8995
20	[2.]	[ 21 369]	[[ 21 369]]	[ 26 372]	369	21	4	6	6	6.24264	6.24264

Figure 5.5: The screenshot of partial primary profile table example. A complete sample contains 16 columns. The information of connected cells, axon root and tips coordinates, general trace location, number of critical node, number of branches, length, etc.

2. **Connected cells:** The unique index(s) of the cell(s) that an axon candidate touches. An axon record can have more than one connected cell. This is usually due to branch entanglement. Cell index 0 refers to the image border for the case of growing out of view, which is another common reason for multiple connected cell labels.
3. **Best connected cell:** The area centre of a neuron cell is assumed to be on the same line as its axon's major branch direction. Abiding with this assumption, an estimation of the best-connected cell can be selected from the connected cells.
4. **All root node coordinates:** The position of the potential root node(s). A root node is a pixel that touches a cell at its boundary or image border. The number of root nodes is greater or equal to the number of connected cells, as there can exist more than one touched point on one cell. This issue will be addressed later in the processing.
5. **Best root node coordinates:** The root position associated with the best connect cell. If multiple root positions exist, we pick the first one.
6. **Leaf nodes coordinates:** The positions of all leaf nodes. Here, a leaf node refers to the tips of an axon candidate, including the growth-cone.
7. **Growth-cone node coordinates:** The growth-cone is the structure leading the growth of an axon. In this system, a growth-cone node is defined as the furthest leaf node in pixel distance to the best root node along the segmented axon trace.

If there are two tips that share the exactly same distance (which happens on very short axons), then we select the one with greater Euclidean distance. If this is the same, then we choose the one with greater distance to its parent cell centre.

8. **Branching node coordinates:** A branching node represents the place where a sub-branch emerges. There are some issues on precisely determining its coordinates. Rules are implemented to resolve these ambiguities. More details can be found in Section.5.2.4.
9. **Number of nodes:** The total number of root nodes, branching nodes, and leaf nodes. For example, if an axon only has the main branch (stem), its number of nodes equals to  $2 = 1$  root node + 1 leaf node. If an axon has one sub-branch, then, the number should be  $4 = 1$  root node + 1 branching node + 2 leaf nodes. The number of nodes in the primary profile is not equivalent to the ultimate number of tree nodes as new types of nodes will be included later.
10. **Block information:** The conventional bounding box representation, including the most left and top coordinates, and the width and height of the smallest block containing the entire axon object. This information will be used when detecting axon breaks at the repair stage.
11. **Total length:** The total number of pixels belonging to an axon candidate.
12. **Stem path length:** The number of pixels that are located on the path between root and growth-cone.
13. **Stem path:** An array of all stem pixel coordinates of an axon candidate.
14. **All path:** An array of all pixel coordinates of an axon candidate.
15. **Axon center coordinates:** The averaged coordinates of all the axon pixels, depicting the general position of an axon candidate.
16. **Grow out of border:** A binary flag, True when axon touches the border of an image, otherwise False. This is as same as having index  $\{0\}$  in the connected cell(s).

### 5.2.3 The Cell Profile

A profile is also built for cell objects. Its attributes include, cell label, connected axon's indexes, the total length of connected axons, the length of its longest axon, cell block information, cell centre coordinates, and a binary label of touching image border.

### 5.2.4 Correctly Determining Unique Splitting Nodes

The system is designed to transform image data to quantified value. Some attributes such as the position of a branching node may cross several pixels on the original images thus the accuracy depends on the original resolution. Furthermore, even though the skeletonisation function helps to sketch out an axon piece in minimum width, ambiguity could still exist since there is no dependency relationship defined on the grid map. In the other words, if two axon pixels stick together like Fig. 5.6(d), it is not clear which one comes first. This may cause a triple branch to be understood as two double branches.

We need to clarify the definition of these nodes with the language that computer can understand. One intuitive way would be stretching the surrounding pixels to a vector, and calculating the number of isolated axon pixels. However, if we make the number of sub-branches equal to the number of connected positive pixels of a branching node, then the situation of Fig. 5.6(a) could cause a miscount condition. If we count the number of independent positive pixel groups, error cases still exist, jeopardising the accuracy of the system. The method of using a greater radius and combining continuous pixels as a single count was also found imperfect, an outlier case has been illustrated in Fig. 5.6(c).

After attempting different methods, it was concluded that an ideal branching detection function should fulfill the following criteria.

1.  $N_{\text{nodes}} = \sum \text{branching degree} + 2$
2. Each branching position can be represented by a single unique pixel.
3. One pixel can only belong to one edge.
4. The root node will only have one child which means it is not a branching node. If there are multiple branches attached with a root node, treat them as different axons by registering the next pixels as their roots.



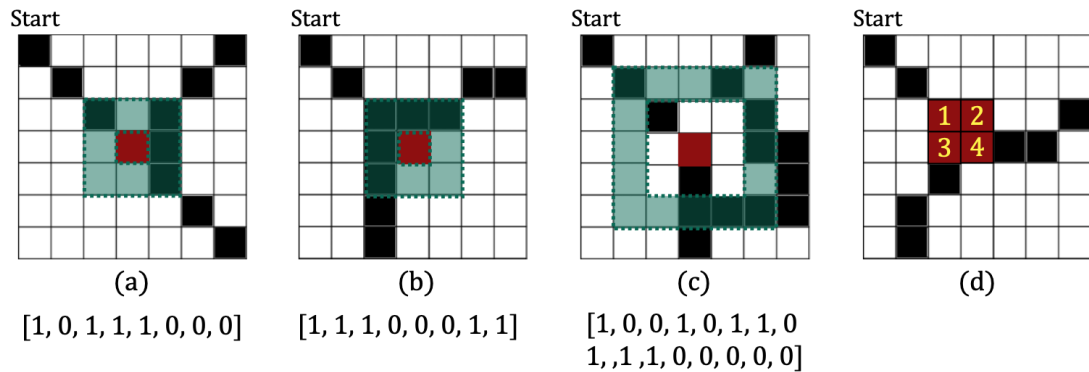


Figure 5.6: The plots demonstrated the ambiguities in determining branching points. All the examples have axons rooted at the left top corner. The number below the grids represent the label of the red node's neighbours. (a) If we define the branching degree as the number of connected positive pixels, then the degree for the red node is 4 which conflicts with the fact of 2 branches. (b) If define the branching degree as the number of independent positive pixel groups, then the given case would report 1 when there are 2 branches. (c) The problem still exists using a larger sampling window. (d) There is a problem of choosing a unique branching position as the four nodes suggested in the example plot all look plausible. The current method which is base on shortest path tracing can resolve these issues, ensuring the branching location determination to be unique and accurate.

In order to meet these requirements, we used a backward tracing approach. The approach traces all the trips of an axon back to its unique root. The positions where different tracing paths meet are determined as branching nodes naturally.

Starting from the furthest tip (growth-cone), the Dijkstra's shortest path method was used to search the axon trace from a tip to its root node as the tracing agent. The path searching method is applied to each tips, and such search stopped once its shortest path touches a pixel already visited position. This first touched node is the position that will be registered as a new branching node. Its child nodes are the proceeding tip and the tip that triggered the touching event. The original parent node will be de-associated with its children and re-assigned to the parental role of this new node. The edge is split by the branching node and appointed to the connected nodes. The process is summarised in Pseudo Code (see Algorithm 2).

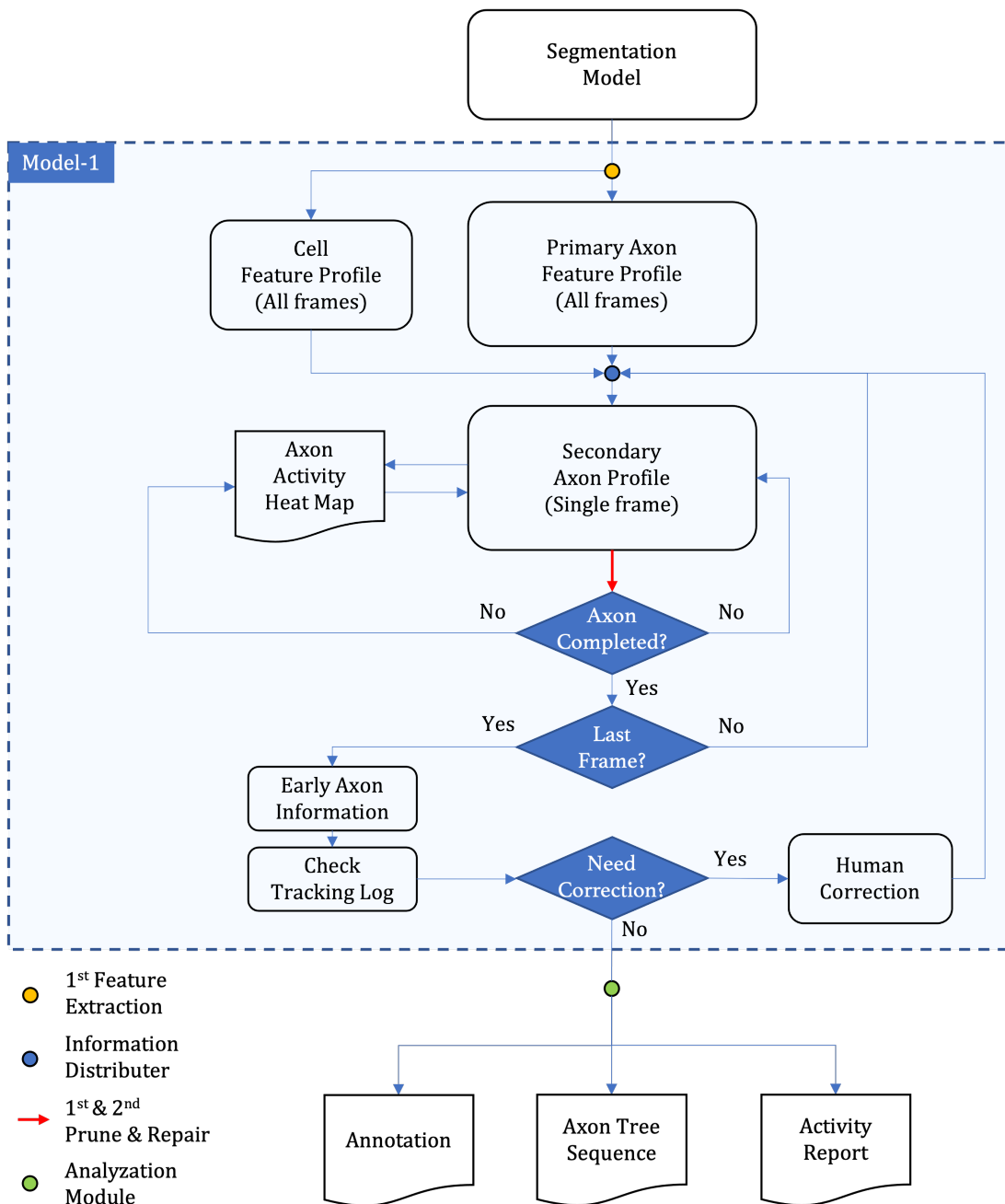


Figure 5.7: The operations and the outputs of the model-1. The primary axon profiles contain crude information collected from segmentation results for each axon candidate, and the secondary profiles are produced on frame basis and structured as a tree. Axon activity maps record the integrated active area of a target object, a visualised example is given in Fig.5.8. Three files are in the outputs, i) axon’s individual-level annotation, ii) raw axon tree files, iii) axon activity table. The procedures for axon fix are simplified in this chart (red arrow). A detailed illustration of it can be found in Fig. 5.12.

**Algorithm 2** Determine Branching Nodes and Get Axon Tree

---

```

1: function GETAXONTREE(tips, root, axonImg)
2:   tips = sort(tips, 'descend')
3:   tree = tree.init(root)
4:   branchingNodes = []
5:   for tip in tips do
6:     path, stopPoint = DijkstraShortestPath(tip, root, axonImg)
7:     if stopPoint == None then
8:       tree.add(node=tip, parent=root, edge=path)
9:     else
10:      parent, child, edge1, edge2 = tree.find(edgeNode = stopPoint)
11:      tree.add(node=stopPoint, parent=parent, edge=edge1)
12:      tree.add(node=tip, parent=stopPoint, edge=path)
13:      tree.edit(node=child, parent=stopPoint, edge=edge2)
14:      branchingNodes.append(stopPoint)
return tree, branchingNodes

```

---

### 5.3 Heat Map Based Tracking Module

The challenge in the axon tracking task is to deal with two crossing situations, when a target axon touches or crosses something axon-alike, and when there is occlusion of the trace of a target axon. To address these, we have to answer three important questions.

1. How to detect such crossing event;
2. How to complete an axon from entangled and broken;
3. How to ensure any new dynamic (e.g. an emerging branch) can be recognised and linked to the right object in the entanglement.

To answer the questions above, the data from a single frame is not sufficient. A very simple example can demonstrate this. If two thinned axon traces are shown to an experienced human annotator and without any temporal context, the annotator would become less confident about the decisions and the annotation accuracy would drop. Even strip-like dregs look like an axon branch when it is close enough to a trace. However, when the annotator is provided several more frames, the decision will be much easier since only axons grow and occluding material is usually relatively static. Entanglements can be separated when the entanglement process is learned.

Therefore, it is critical to find a proper method that can encode an axon's activity history and present enough contrast between an axon target and other structures. We found that this can be achieved using a heat map based method.

The axon tree construction logic can be split into two atom operations, we refer to as Prune and Repair. Prune means checking all affiliated nodes and removing the irrelevant ones. Repair refers to searching procedure, merging the confirmed piece(s) from nearby with the main tree (target axon).

### 5.3.1 History Encoding

A heat map is a classic method of encoding temporal information. Besides the role of providing reference information for axon trace completion, it can also be used directly in the axon activity study as part of the final report. The frequencies of an axon's growth activities as a response to an environmental factor will change in different controlled experiments. This frequency change can be read from a heat map.

An axon heat map synthesises the active areas from a short stack of history maps. The history depth is typically set to  $n = 5$  frames in this system. Once adding a new record to the stack, the record from the earliest one will be removed. These histories are synthesised with weights, updating is executed before loading the next video frame. Greater weights are given to more recent records. There is a balance between the contribution from different times, achieved by controlling the weight decaying speed for both the overwriting and the lagged updating effects. A typical set of weights for  $n = 5$  is  $\{0.4, 0.3, 0.1, 0.1, 0.1\}$ .

$$P_{history}(x, y) = HeatMap(x, y) = \sum_{i=1}^n w_i T_{t-i}(x, y) \quad (5.3)$$

A conventional heat map can reflect the active area of axons as a type of object, but is not able to differentiate individuals. As our approach is semantic-based, a tertiary-value transformation is utilised to encode contrast into  $T_t$ . For a position  $(x, y)$ , its label is valued as,

$$T(x, y) = \begin{cases} 1 & \text{Target Axon} \\ 0 & \text{Background} \\ -1 & \text{Other Axons} \end{cases} \quad (5.4)$$

An axon piece not connecting to any cell and located on a generally positive area in history is more likely to be a part of the target axon, thus encouraged for building a connection, vice versa. Value 0 is given to the background to erase activity signals. If an axon has left an area for enough time, the heat map value on this position will

gradually return to zero. If was occupied by another axon, the area will need a short period of cooling down.

To curb the risk from segmentation error and avoid a decision cliff a tertiarised map is dilated by a  $5 \times 5$  kernel, and then smoothed by a  $5 \times 5$  ( $\mu = 0, \sigma = 2$ ) Gaussian kernel. An example of a heat map is shown in Fig. 5.8. Given a case of two axons entangling together, the negative-aversion effect will help separate the traces where the approximate area previously belonging to the non-target axon is recorded. The dilation effect and the neutral background ensures the bud after a target axon breaks through another trace still can be detected. With a history encouragement and aversion prior, a context-guided repair/prune decision can be made at the pixel level.

When considering cutting a branch, its score is estimated by averaging all the positions within it. For example, there is an axon branch  $b$  constituted up by  $l$  pixels, the overall score of this axon is computed by,

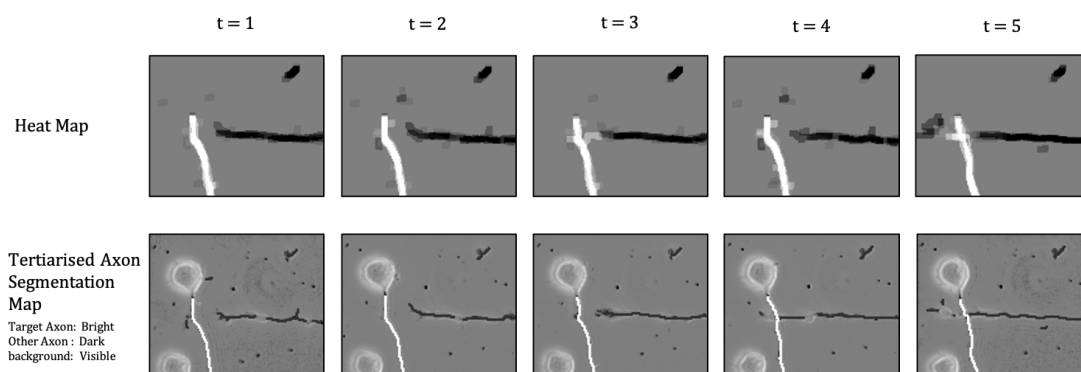
$$P_{heatmap}(b) = \frac{1}{l} \sum_{i=1}^l P_{history}[bx_i, by_i], \quad (5.5)$$

where  $(bx_i, by_i)$  represent the pixel coordinates on branch  $b$ . The prior information is learned from history synthesis, but the affiliation likelihood on the current frame is still unknown. We still cannot tell whether an axon should be merged with a nearby piece that was not recognised. The other morphology-based features are introduced to the decision equation. Combined with  $P_{history}$ , the ultimate decision is ready to make.

To initialise a heat map, the system requires the user to provide a stable frame where the axon piece to be completed is identified. The history stack is filled with  $n$  copies of the initial map  $A_0$ . This requirement is easy to meet, as in practice, an axon target is very small (several pixels) when first emerging from a cell. For the case of starting from a long and broken trace, we selected a later frame that is complete and then execute the repair /prune and tracking algorithm on the reverse direction until the target frame is reached. The axon images that have been processed are used for generating the heat map instead of the original initialisation.

### 5.3.2 Task 2 Solution: Complete Axon Trace with Two-Stage Checks

An axon branch can be read from a heat map. A crossing event happens where the score plunges. We have been able to answer the first of the three listed questions (beginning of Section.5.3) in general. We now focus on the second question of axon trace completion. It can be separated into two operations of repair and prune.



(a) Heat map decomposition

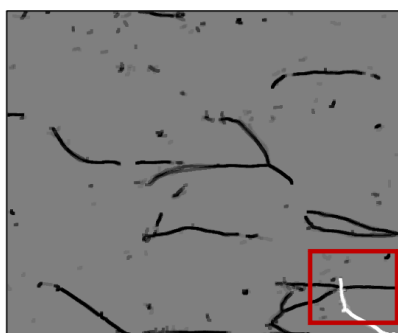
(b) The entire heat map at  $t=21$ 

Figure 5.8: A heat map synthesises a stack of tertiary-valued maps with decaying weights. In the first row plots of (a), the majority grey represents 0, the neutral value of the background. And the area with a lighter colour (positive values) is where the target axon was laying in a collection of  $n = 5$  history. Similarly, a darker colour (negative values) indicates the activity area of other axons. The candidate located in a positive area is preferred and has a higher likelihood to be accepted by the main tree. (b) The entire heat map, where the red rectangular scoped the illustrated area of (a). The heat map can discriminate complex crossing elements and maintain the completeness of the tracked object.

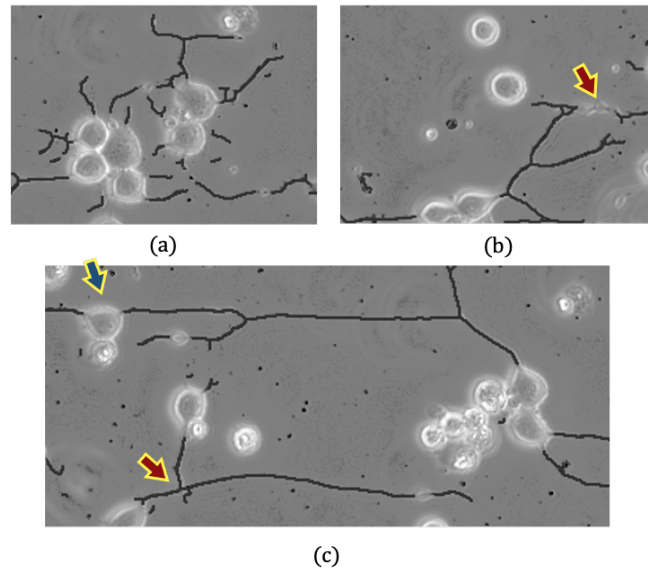


Figure 5.9: A branch can be disconnected or misconnected for many reasons. A major goal of the model-1 is to find and fix these problems without regression. The following cases are some of the problems observed during development. In the plots above, the axon areas are marked by dark lines which are not their natural colour. Morphological fixation cannot be too aggressive, otherwise, misconnection of axons and branches loop may occur, such as the situation demonstrated in (a). However, this inevitably leaves some greater gaps unfilled, such as the break in (b). The green arrow points to a case where the trace is overlapped by a cell. The model-1 has to extract history attributes to get the essential information for reconnecting them. The red arrow displays the challenge of identifying new dynamics after axons touched, in which the vertical branch's tip is emerging from the other side of the horizontal branch.

Before going into the solution, consider the conditions that can potentially break an axon. The typical reasons learned from the data study are, 1) segmentation error, 2) poor illumination, and 3) obstruction. In a binary axon map, with background factors removed, all the breaks are in the same form presented as isolated pieces near an object. The 'easy' breaks have been corrected by the dilation-erosion fixation executed in Section.5.2.1. The potential breaks that still exist on maps would have a greater gap to overcome (Fig. 5.9 (b) and (c)).

For the model-2, as in an instance segmentation (e.g. Faster RCNN), the low-level features extracted by the middle layer of the segmentation model can be used. A repair decision depends on the axon's heat map scores and its morphology information.

Assume we have had a repair decision function  $\Omega(a_t, a_c)$ , in which  $a_t$  is a target axon and  $a_c$  is a candidate that matches  $a_t$ . The most intuitive way might be repeating  $\Omega$  on every combination of targets and candidates, and the complexity would be  $O(n^2)$ .

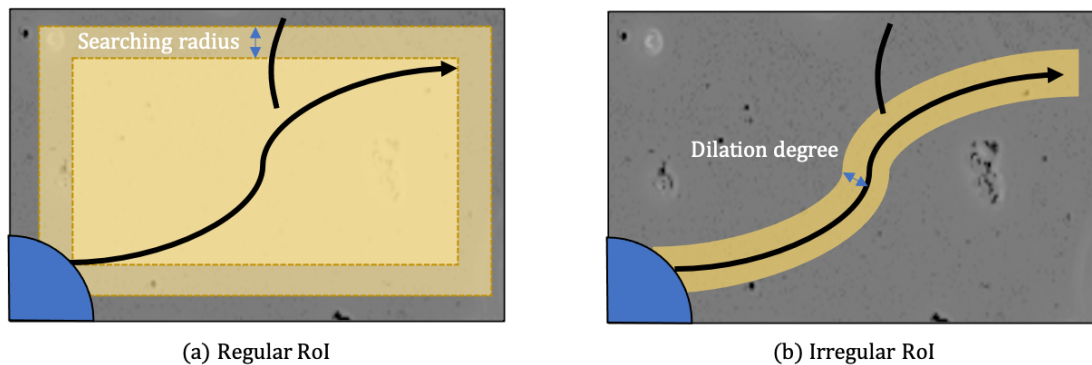


Figure 5.10: The two sorts of optional RoI windows for axon rough searching. Demonstrated by (a), a regular RoI scheme is block-based, while the (b) irregular RoI is area dilation based. Both schemes have an allowance margin controlled by the searching radius  $r$  and the dilation iteration number  $n$  respectively.

This can be costly when a larger camera view has tens of objects. A two-stage search plan is used to replace the greedy search, known as candidate Rough Search and Candidate Formal Search. Function  $\Omega$  is also broken into  $\Omega_{RS}$  and  $\Omega_{FS}$ . During the rough search stage, the irrelevant majority is removed from the candidate list. Then, in the formal search, a target-relevancy score will be computed.

After the affiliation relationships are determined, any breaks will be repaired using an optimal trace algorithm to estimate the missed part, which is based on the Dijkstra's shortest path. More repair algorithm details can be found in Section.5.3.4.

### Rough Search

A Region of Interest(RoI) window is used in the rough search, the indexes of included axons will be added to a candidates list. The region of interest can have either a regular or an irregular shape.

For the regular RoI function, a rectangular window is selected. Its position and scale are estimated by the target axon's 'block information' in the primary profile (see Section.5.2.2) plus a searching radius which is a user-adjustable parameter. For example, if the "block information" of an axon is ( $top = y, left = x, width = w, height = h$ ), image size is  $(W, H)$ , and the user-defined searching radius is  $r$ , then the remaining candidates must be included by the block of following, where maximum and minimum operation is used for avoiding searching block leaving image scope.



$$RoI_{reg}(x, y, w, h, r) = \begin{cases} top : \max(y - r, 0), \\ left : \max(x - r, 0), \\ width : \min(w + 2r, W - x + r), \\ height : \min(h + 2r, H - y + r). \end{cases} \quad (5.6)$$

For irregular-shaped RoI, a searching window is generated by axon mask dilation function  $D$ . Similar to the parameter  $r$  in the regular approach, a dilation degree  $n$  is defined for controlling the searching space. It represents the number of times a  $3 \times 3$  kernel is applied, shown as,

$$RoI_{irr}(mask, n) = D_{3 \times 3}(mask, n) - mask \quad (5.7)$$

A regular window is computationally efficient but is more susceptible to noise as it does not use an adaptive searching area. It will be appropriate for sparse objects, or axons are vertically/horizontally distributed. On the other hand, an irregular window is adaptive and is desired when the camera field is cluttered. In practice, as the experiment is working with chemical patterns (more information about this topic is in Chapter. 6), axons are guided to grow either following or repelling chemical cues. The option of a regular window is more frequently used as it is set as the default configuration.

The objects that are left in the RoI are the candidates ready to be connected. The list will then go through three checks and any invalid candidates will be removed. This first sift is very loose since it is only for filtering the obviously invalid objects.

1. A candidate piece must not connect to a cell;
2. A candidate piece must not have been marked as 'to be pruned' in a manual correction order;
3. The averaged heat map score of the entire candidate piece must above the threshold  $t_{rough}$ .

The steps are illustrated in Fig. 5.10. To summarise, the rough search function takes three parameters, window scheme (regular/irregular), search degree(block increment radius/dilation degree), and minimum averaged heat map score  $t_{rough}$  (by default  $-0.1$ ). The inputs are the axon probability map and axon profiles. It returns a list of candidates (valid axon pieces in the scope).

**Algorithm 3** Rough Search

---

```

1: function ROUGHSEACH(profiles, windowShape, searchDegree, t)
2:   axonsMap = zeros(profiles.imageSize)
3:   for i, area in enumerate(profiles.axonArea) do
4:     axonsMap[area] = i+1
5:   if windowScheme == 'regular' then
6:     mask = getBlock(profile.block, searchDegree)
7:   if windowScheme == 'irregular' then
8:     mask = dilate(profile.axonArea, (3x3, searchDegree) - mask)
9:   candidates = unique(axonsMap[mask])
10:  candidates.remove(backgroundIndex)
11:  candidates.remove(targetAxonIndex)
12:  for candi in candidates do
13:    if exist(candi.cell) or heatMap(candi).mean < t or banned(candi) then
14:      candidates.remove(candi)
return candidates

```

---

The rough search step aims to find all the potential pieces. A candidate at this stage will have a chance to be merged but this does not necessarily mean it is a part of axon. More information of a candidate will be gathered in the formal search.

**Formal Search**

After the Rough Search a preliminary candidate list is created. Not much is known about them except they are close to the target axon. These candidates will be merged with the main tree later in the process (see Section.5.3.4). In some cases, if a tree's structure is changed, the information on each of its nodes will also be updated.

The formal search collects the essential features for axon nodes, and adjusts the position of nodes and the length of edges.

Factors other than history are also considered in the formal search, including the estimated path linking the target axon and a candidate axon break. Besides that, formal search rectifies the structure of the input axon. In order to provide a more precise measurement, the long edges in the tree are divided into smaller pieces under a maximum edge length threshold.

Due to this operation, there are four types of nodes, root, tip, branching node, and breaking node generated by splitting long edges.

This logic is also suitable in the opposite way, shown in Fig. 5.11. Multiple rounds of prune and fix can generate some very small edges. These small edges can be combined with their consecutive edge by canceling the middle breaking nodes. Imagine if

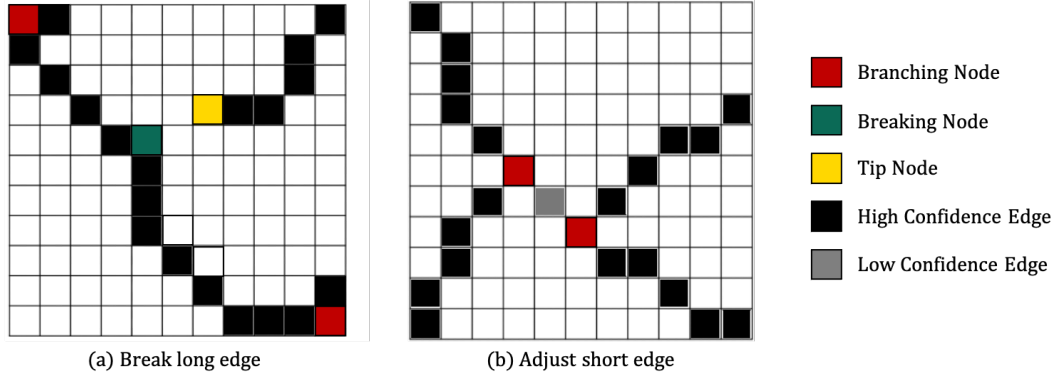


Figure 5.11: A demonstration of the edge adjustment motivation. (a) A broken branch is repaired by connecting one of its tips to the best-matched node on the target axon. Increasing the number of nodes can provide more options and improve the repair quality. (b) Many reasons can lead to a very short edge, where a single low-confidence pixel can lead the entire branch to be disassociated from the target axon. Merging or considering an edge that is a blow a minimum length level with a consecutive edge can help mitigate the risk of this.

the connection confidence of a single pixel is low but the position should not be cut, this combination can salvage the branch. In the case where an edge is under the minimum length threshold but not able to be combined with a neighbour edge (no breaking nodes on both sides), then calculate its scores with its prior edge.

Five scores will be generated and assigned to the nodes as attributes, including  $S_{major}$ ,  $S_{noise}$ ,  $S_{path}$ ,  $S_{stem}$ , and edge length. The definitions of these are,

$$S_{major} = \frac{N_{path}^+ + N_{estimate}^+}{N_{path} + N_{estimate}} \quad (5.8)$$

$$S_{noise} = 1 - S_{major} \quad (5.9)$$

$$S_{path} = \frac{\sum_{i=1}^{N_{path}} p_i + \sum_{j=1}^{N_{estimate}} p_j}{N_{path} + N_{estimate}} \quad (5.10)$$

$$S_{stem} = \frac{N^{stem} + N_{estimate}^{stem}}{N + N_{estimate}} \quad (5.11)$$

$$S_{direction} = \theta_{parent} - \theta_{child} \quad (5.12)$$

The variables  $N_{path}$  and  $N_{estimate}$  refer to the number of pixels on the candidate's

path and its estimated connection path, and  $N^+$  is the number of the pixels with a positive heat map score.  $S_{\text{major}}$  and  $S_{\text{noise}}$  represent the portion of positive scored pixels over the entire trace of a candidate.  $S_{\text{path}}$  is the averaged heat map score of all pixels.  $S_{\text{stem}}$  counts the portion of the stem pixels (the pixels on the path from growth-cone to root node) of an edge. Using an additional stem score can help ensure the completeness of main branches, which is a major objective.

---

**Algorithm 4** Formal Search
 

---

```

1: function FORMALSEARCH(candidates, heatMap, targetAxon)
2:   for candi in candidates do
3:     targetAxon = merge(targetAxon, candi, edgeMax=10, edgeMin=5)
4:   for node in targetAxon do
5:     parent = node.parent
6:     node.edgeLength = len(node.edge)
7:     node. $S_{\text{major}}$  = len(node.edge;0)/node.edgeLength
8:     node. $P_{\text{noise}}$  = 1 - node. $S_{\text{major}}$ 
9:     node. $P_{\text{path}}$  = mean(heatMap[node.edge])
10:    node. $S_{\text{stem}}$  = len(node.stem)/node.edgeLength
11:    node. $S_{\text{direction}}$  = parent.direction - node.direction
   return targetAxon

```

---

The scores are assigned to all input tree nodes as object attributes.

Formal search can be understood as a feature extraction module, and is used as a tool in the rounds of repairs and prunes for updating attributes, after which different thresholds are applied. It is replaced by the multi-layer perceptron in a Graph network based model-2 (see below).

### 5.3.3 Task 3 Solution: Iterative Connect&Cut

The rough search sifts the candidates to be merged with the main tree. The formal search generates the attributes for nodes. The features recorded at a node, as well as information from either parent or child nodes used in the node classification. The axon completion task is turned into a graph separation task.

It is not a convex problem, since between a high and low  $S_{\text{path}}$ , either positive and negative instances could exist. To find the border between instance clusters, an effective old school solution was chosen, the node classification will follow a bottom-up strategy just like decision trees.

The input of this decision model is node instances. Depending on the operation, the output label can be either repair/don't-repair, or prune/don't-prune.

The node instances have to be processed with an order. The dependent relationship between parent and children nodes is a decisive factor. A negative flag (not repair or prune) can transmit along the descendent direction. Once a prune decision is produced, all its succeeding nodes will be disassociated with the target tree, so do the not repair decision.

Instead of abandoning all nodes from the disconnected position forever, the repair&prune will be executed for three rounds, as shown in Fig. 5.12. The nodes are scanned in turn, thus it is possible to reverse a decision. Each round is executed with different factors, the threshold parameter, both the inputs and the attributes attached to the inputs, and the order of processing instances. There are intentions behind designing each round, which be explained in the next two sections. This will help explain the model and the choice of parameters.

### **First Round**

After the rough search, a list of possible nearby pieces is generated. The role of the first round is to fix these pieces and make a preliminary cleaning.

A nearby signal can be a combination of breaks and noise. Its closest node to the target axon is not guaranteed to be true, vice versa. Basically, there are two possible options, either repair-first or prune-first in this round. The repair-first runs in the first round.

The prune-first scheme can protect the main tree from being polluted, as involving wrong branches would produce an incorrect activity heat map thus diverting tracking to an irreversible situation. But a small prune mistake in one frame can also make the system lose track in the frames afterwards as the decision also lowers the heat map confidence in this area. This was more commonly encountered in practice. This is double-edged. The decision with the prune-first principle would bear more risk because the prune-first decision is made based on a subset of the completed axon tree.

The repair-first scheme uses a greedy approach, and is more conservative as it ensures the connection of possible materials (axon nodes). There is no rigid computation requirement for this task. The repair-first scheme has its complexity in a user-acceptable range.

In order to avoid missing a true break, the repair filtering is comparatively looser than the first pruning (not much beyond the rough check requirements), where a break just needs to be non-cell related, and have  $S_{noise} < 0.3$  and  $S_{path} > 0.2$ . These are the proper thresholds acquired from parameter searching. At the pruning stage, the axon tree will be re-sorted and updated by a formal search, this is to fix the length

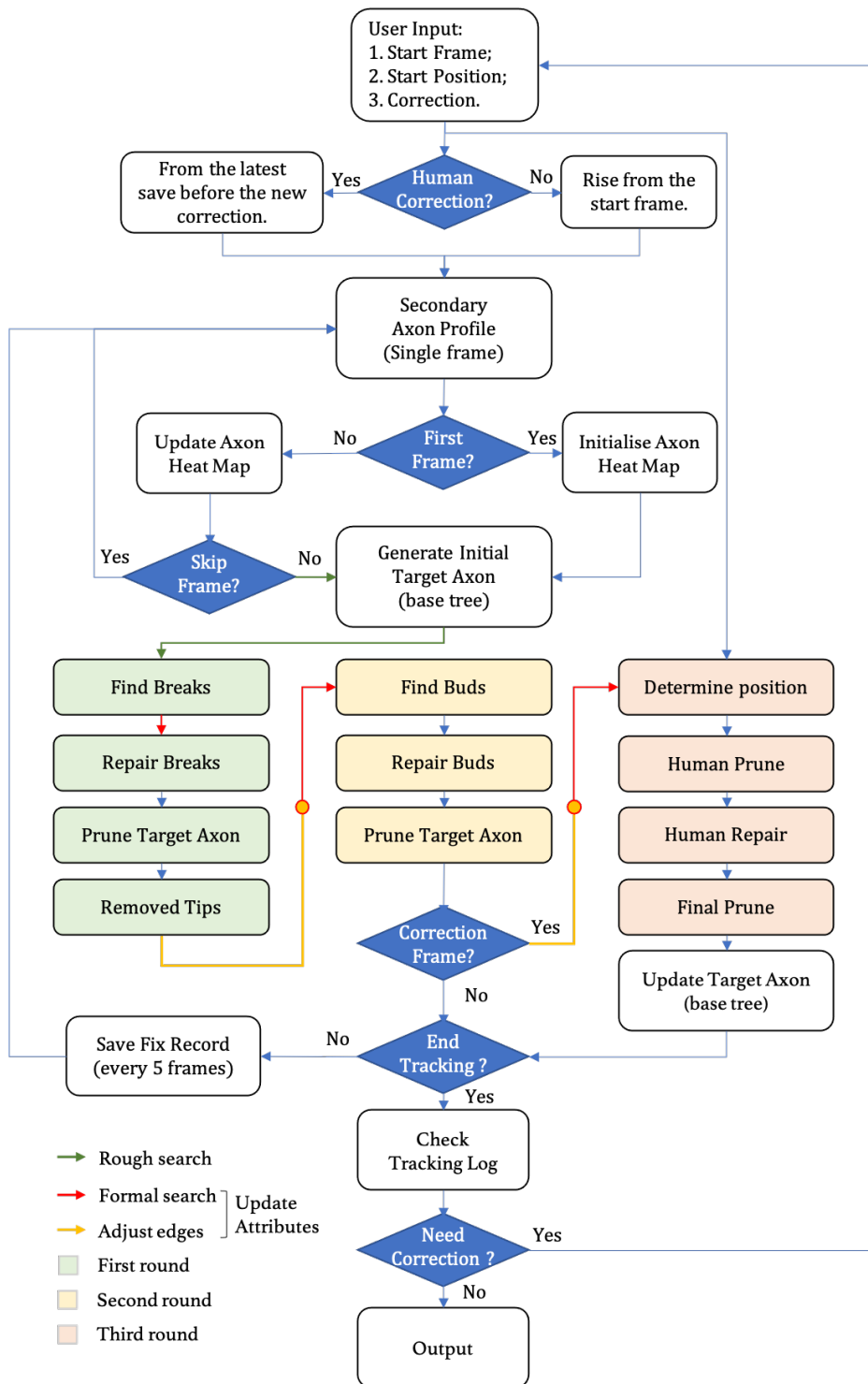


Figure 5.12: The three-round repair and prune procedures in the model-1. The objectives of the three rounds can be summarised as, 1) general correction, 2) bud correction, and 3) human correction, respectively. The rounds are indicated by the boxes in different colours, and the process of two searching stages (rough and formal) are simplified and represented by green and red arrows. A detailed look of the function 'Generate Initial Target Axon' can be found in Fig. 5.15 which creates the initial main tree in a frame.

of edges and combine the very short ones. Checking all the nodes, removing those with  $S_{path} < -0.1$  and adjusted edge length to a maximum of 10 units (pixels, can be different for images in other resolutions). This is because a lower edge length limitation can provide more nodes for building links.

At the pruning stage, the strategy changed. A node with reasonably high confidence will not be considered if its parent node is classified as negative. So for  $S_{path} < -0.1$ , it can be strict and too sensitive for small edges when splitting entanglements. The solution is slightly increasing the length configuration. For the situation that the target axon is cutting into another axon, a longer tip edge can help protect the branch from being pruned, as the negative impact of its average heat map score is diluted. For the situation of another axon cutting into the target axon, longer intermediate edges can avoid accidents. But this will not impede the removal of irrelevant branches as their edges are from branching nodes and the length adjustment is working on breaking nodes. Contrarily, a longer bridging edge (from the parental branching node to the first node on the sub-branch) will dilute the positive near-stem impact and help detect those noise nodes.

Generally speaking, the length control for repair and prune are different in the first round.

### **Second Round**

The second round aims to retrieve any true nodes eliminated in the first round by accident, especially bud nodes. A bud refers to a group of branches having a tip node and a short edge (Fig. 5.9 (c) red arrow).

In practice, though most axons have been very clean, small branches such as emerging buds are frequently recognised as noise. Such delicate growth dynamics are critical for studying axons' chemical-cue response. Misclassification of them could lead to a delay in growth speed screening. More specifically, most of this type of misclassification happens at the first frame after the target axon crossed something, e.g. another axon, and just emerged from the other side.

The bud tip would always have a senior node on its branch to be clearly negative on the heat map as the crossing area was occupied by another axon.

There are two situations of negative samples from the first round needing to be checked for finding the buds.

Case 1: Joint disconnection due to the decision on a senior node, and

Case 2: Disconnection due to low scores on itself.

The pruned nodes will be sorted into different lists according to the cases. Following the definition, all the buds should come from case-1. The second round cut includes and analysis of the tips (leaf/end nodes).

The bud filtering is based on a cost function;

$$C_{connect} = 1 - (w_1 S_{path} + w_2 S_{noise} + w_3 S_{stem}) \times \Delta_{direction} \quad (5.13)$$

Not only do we remove the noise tips but we also remove the buds belonging to another axon if the tips of two axons jet meet and are crossing each other. The direction is considered. All the tip nodes will be compared with the removed nodes from the first round. Buds are assumed to continue growing in a straight line, thus their direction should be consistent with the previous growth in some degree.

If the direction is close to an eliminated branch in the last frame, a negative factor should be included in the consideration. In conclusion, If a case-1 tip node passed the Eq.5.13 check but was found to be close in both position and edge directions to an earlier remove decision, as well as is under a floor level of  $S_{path} < 0.9$ , these nodes will be flagged again to be removed.

### Third Round

The last round is the final part of the completion of an axon tree. It has two procedures, human correction and the final prune.

First is human correction, which is an essential stage under the current framework. Without it, the model-2 will not be able to outperform the teaching model. The model-1 has high accuracy when fixing the main branches but sometimes makes mistakes on small sub-edges. There are three keywords about this step, *when*, *how*, and *which* for the three correction tools, listed as repair, prune, and skip.

The *when* refers to the timing of intervention. As shown in Fig. 5.12, the system will always save a copy of the axon object including the tree and all node attributes automatically every 5 video frames by default. Human correction is applied after a preliminary run of automatic tracking. The procedure will generate a tracking log in video format. Its length depends on the lifecycle of a target axon, normally around 2-5 frames.

A user can select the position (only for repair and prune) and the frame that needs a correction by watching the log, then start the system again with these new inputs (see Fig. 5.13). The correction tool will help the user to find a candidate for connection, selecting the closest node of an unaffiliated tree on the map according to the input position. Instead of repeating the whole computation, the system will be re-started



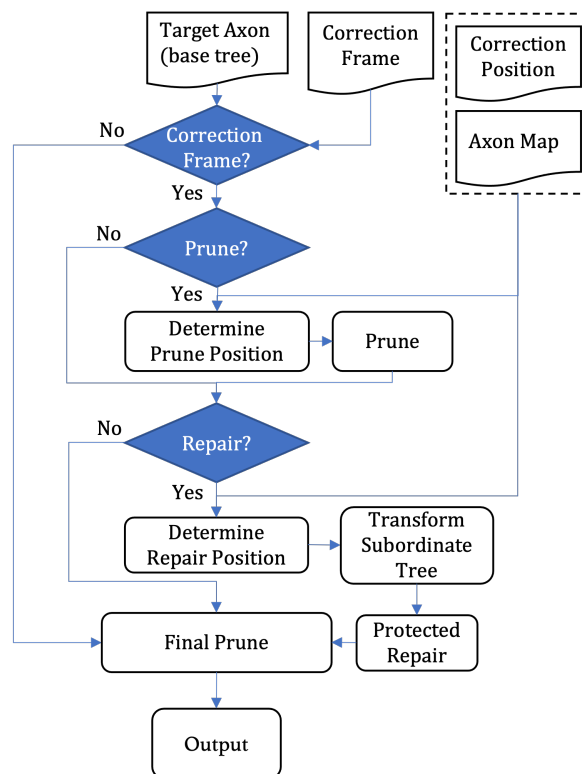


Figure 5.13: The process of manual repair and prune operations, which are parts of the third round. One needs to input the correction frame index as well as the operation type and position. As the complement of the model-1, the knowledge of human correction is integrated into the axon annotation and will be learned by the model-2.

from the latest auto-save point before the correction frame. If it is the frame with a skip comment, the tracking process of the frame will be stopped at the beginning. For manual repair and prune, the correction operations will be applied onto the main tree after the bud checking step which is the end of the second round. At the next auto-save point, the human correction information will be saved together with the tree information. And the previously saved correction will be printed before each run to remind user about the changes.

As for *how*, the repair and prune are executed generally similar to the repair and prune decision from the model itself. The differences are, (i) a human correction decision has a higher priority which cannot be modified by system decision, and (ii) a connecting node is designated. This means once a manual repair has made a connection between a main tree node and a subordinate tree, the system cannot prune that closest node which is picked by the user, neither by cutting it directly nor cutting a parental node of it. This rule also covers the case of manual prune. The outcome of human correction is protected. Skip is the last resort of correction. For the very rare cases that cannot be fixed by any means, a user can call skip to jump over the tracking on this frame, the skip tool will also copy the record of the previous frame and attach it after the tracking list.

The execution order of tools is critical. The skip tool is called before the other two thus has the highest priority. The prune step is executed before repair. The reason behind this rule is relevant to the repair logic which will be introduced in the next section. If we make repair before prune, it would be possible that an early repair decision connects the subordinate tree to an error node on the main tree, which is going to be pruned. Then the pruning rule will be contrary to the protection rule. In conclusion, the priority of execution are 1) skip, 2) repair, and 3) prune.

If there is no human correction comment in the input, then the system will pass the main tree to the final prune function directly. Or, if there is any correction comment, it will execute the final prune after the manipulation. The final prune is a check after the manual repair to trim the subordinate tree in the case of any irrelevant nodes attached, and double-check the quality of the fix operations. Its setup is the same as the first round.

The entire model-1 including all the parameters is tuned on our dataset and has been tested with more than 127 videos on hundreds of axon objects. Overall, the model-1 plus minimum human supervision is able to achieve satisfactory tracking results. The data produced during this process has reached the accuracy level for required for

experimental analysis. A comparison between system-generated data and that from a manual analysis records is given in Chapter 6.

### 5.3.4 Repair Mechanism

The process of repair is made up of merging trees. In the previous sections, the decision-making of repair and prune has been explained. Also, we have mentioned that the shortest path will be used as an estimation of the missing part between a break piece and the target axon. But there can be more than one candidate in the list, and many nodes on the main tree can be connected. It is important to consider the order of the merging operations.

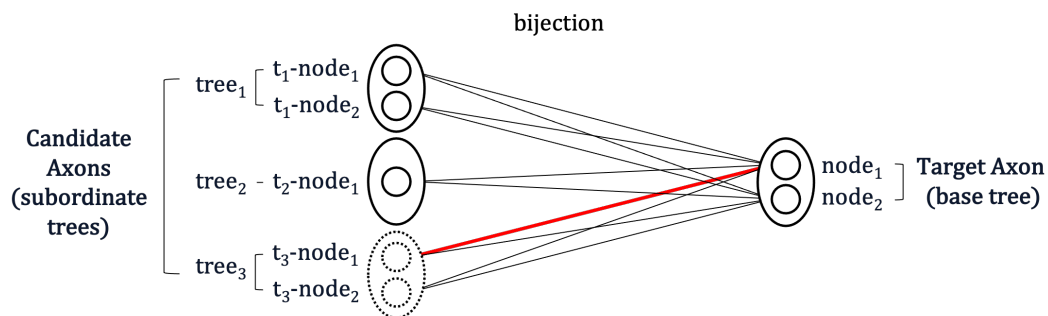
Our method has been shown in Fig. 5.14, where a bijection connection matrix will be generated. The matrix columns represent the nodes of the basic tree and the rows represent the nodes of the candidate trees that are selected in each round. Several candidate trees can be ready-to-connect at the same time, having a total number of  $n_{candi} \times n_{target}$  number of choices.  $n$  here is the number of nodes. Every entry depicts the repair cost between the column indexed node to the row indexed node. Axon repair is an iteration-based task, having one row executed each time. By picking the minimum value in the matrix, the optimal repair pair at the time can be determined.

Instead of treating all the repairs in parallel which should not work, the iterative way can reconstruct the dependent relationship between candidates. For example, imagine a branch that is cut into many small pieces. To repair it, one would expect to start from the closest one instead of treating them equally.

This means any change in the main tree (target axon) will influence the connection in the next operation. The matrix has to be updated after each iteration. Once a node is selected, all the rows from the same tree will be removed from the matrix (the black arrow in Fig. 5.14). These nodes are registered as new columns and will join the connection competition but on the other side in the next iteration. For a row in the matrix, its minimum score may decrease after an update when a new available node is approaching them, but will not increase.

Selecting a row node means the repair decision of the entire candidate tree is made. At the beginning, a root node is initialised arbitrarily from all valid root positions. The reason of having multiple potential root positions has been explained in Section.5.2.2. But before connecting it to the main tree formally, the candidate tree has to be transformed to the form of having the selected row node as its root.

This root will become a child node of the column indexed node from the main




			Target Axon (base tree)			
			node <sub>1</sub>	node <sub>2</sub>	node <sub>3(new)</sub>	node <sub>4(new)</sub>
Candidate Axons (subordinate trees)	tree <sub>1</sub>	t <sub>1</sub> -node <sub>1</sub>				
		t <sub>1</sub> -node <sub>2</sub>				
	tree <sub>2</sub>	t <sub>2</sub> -node <sub>1</sub>				
	tree <sub>3</sub>	t <sub>3</sub> -node <sub>1</sub>	Optimal			
		t <sub>3</sub> -node <sub>2</sub>				

Figure 5.14: A visualisation of the repair process. The potential connections are generated by a bijection between all candidate tree nodes and all main tree nodes. As the repair is executed in an iterative manner, only the optimal pair will be processed in each iteration, which is the one with minimum cost. A candidate tree will be locked when any of its nodes was selected. After a repair step, the matrix will be updated by removing the selected tree and adding new options to the main tree side. The connection costs are measured by Eq.5.14.

tree, and the edge in between is estimated by a Dijkstra's path searching agent. The opposite segmentation probability map ( $1 - p$ ) is used as the cost searching reference in this step. The heat map is not used because it is not as accurate as the segmentation map regarding the trace position estimation at the time.

The selection procedure will continue until the minimal entry of the matrix is found greater than a preset minimum threshold.

About the matrix, the connection cost is mainly contributed by the path cost on the segmentation map. But the distance is not the only factor in this selection. From trees to trees and nodes to nodes, the differences in their semantic attributes will also influence the selection. The cost function is

$$S(n_i, n_k^{tree_j}) = cost(n_i, n_k^{tree_j})(1 - S_{path}(n_i, n_k^{tree_j})) - L(tree_j) * w_j - 10w'_{j,k} \quad (5.14)$$

The connection cost contains 5 parts, which are

1.  $Cost(n_i, n_k^{tree_j})$ , The lowest path probability cost between  $(n_i, n_k^{tree_j})$ , the  $i$  node on the main tree and the  $k$  node on  $j$  candidate tree.
2.  $S_{path}$ , the averaged heat map score along the estimated lowest cost path.
3. The tree length factor of the  $j$  candidate tree,  $L(tree_j)$ . The longer break will have a higher priority to be connected. It is reasonable as a longer piece normally is also more value for analysis.
4. The tree bias weight of the  $j$  candidate tree, represented by  $w_j$ , which depends on the scenario. By default, it is set to the mean of overall averaged stem score and path score, as  $(S_{stem_j} + S_{path_j})/2$ .
5. The node bias weight of the  $k$  node on  $j$  candidate tree, written as  $w'_{j,k}$ . By default it is set to be  $S_{path_{jk}} + S_{stem_{jk}} - S_{noise_{jk}}$ .
6. A selection threshold factor  $t$ . A higher threshold leads to a stricter process and more candidates will be left as un-connected.

A repair operation can potentially cause double-count issues on existing edges, as the shortest path on the reference map may go through the target axon. Therefore, any new part will be scrutinised after being repaired. Additional nodes will be registered at the overlapped positions. This happens before the edge adjustment introduced in

Section.5.3.3. If two nodes are too close, one of them will be relocated or cancelled accordingly.

### 5.3.5 Determine Initial Segment, Cell and Root

The basic logic of the rule-based model is finding a most-matched initial segment (the initial main tree), completing it, and adding it to the tracking sequence. The initial segment in a frame is decisive for the whole process. A good model can determine it accurately preventing the model from shifting the tracking focus to another trace nearby. A general illustration of the scheme is in Fig. 5.15

An essential character of the main tree is it should have a relationship with the parent cell. Since an axon is short at the beginning stage, typically from 2 to 5 pixels, greater relative-length change is a challenge for the model to make a decision based on the other ratio scores.

Usually, the neuron cells in the same area tend to make similar responses to a biomedical cue in the area. Here a response can be either a movement or cell membrane dynamic. And consequently, the affiliated axons grow oriented in a small range of directions. In this case, one can easily occupy the area that was belonging to another on the heat map. Besides that, cell movement can erase the axon signal from the position it was located.

These two reasons suggest that the root and area history are not sufficient for finding an initial segment. The problem can be easier by finding the parent cell first. As the position of a cell is relatively constant, we used a heat map approach to track neuron cells from frame to frame. Unlike the axons, a cell will not vanish. Except the border cases, one can take the candidate with the greatest score as the next object without setting a minimum rule-out threshold.

To cover potential segmentation errors, especially those due to the bright cell rim, the best-matched axon may not actually touch the right cell. The model will increase the margin allowance gradually to reach the best match axon candidate.

The four steps of finding the succeeding trace of a target axon in a new image are;

1. Given the position information of the completed target axon in the previous frame, apply a rough search to obtain a list of axon candidates for matching.
2. By comparing the cell activity heat map, find out the corresponding cell object in the current frame. This is the succeeding parent cell of the target axon in the current frame.

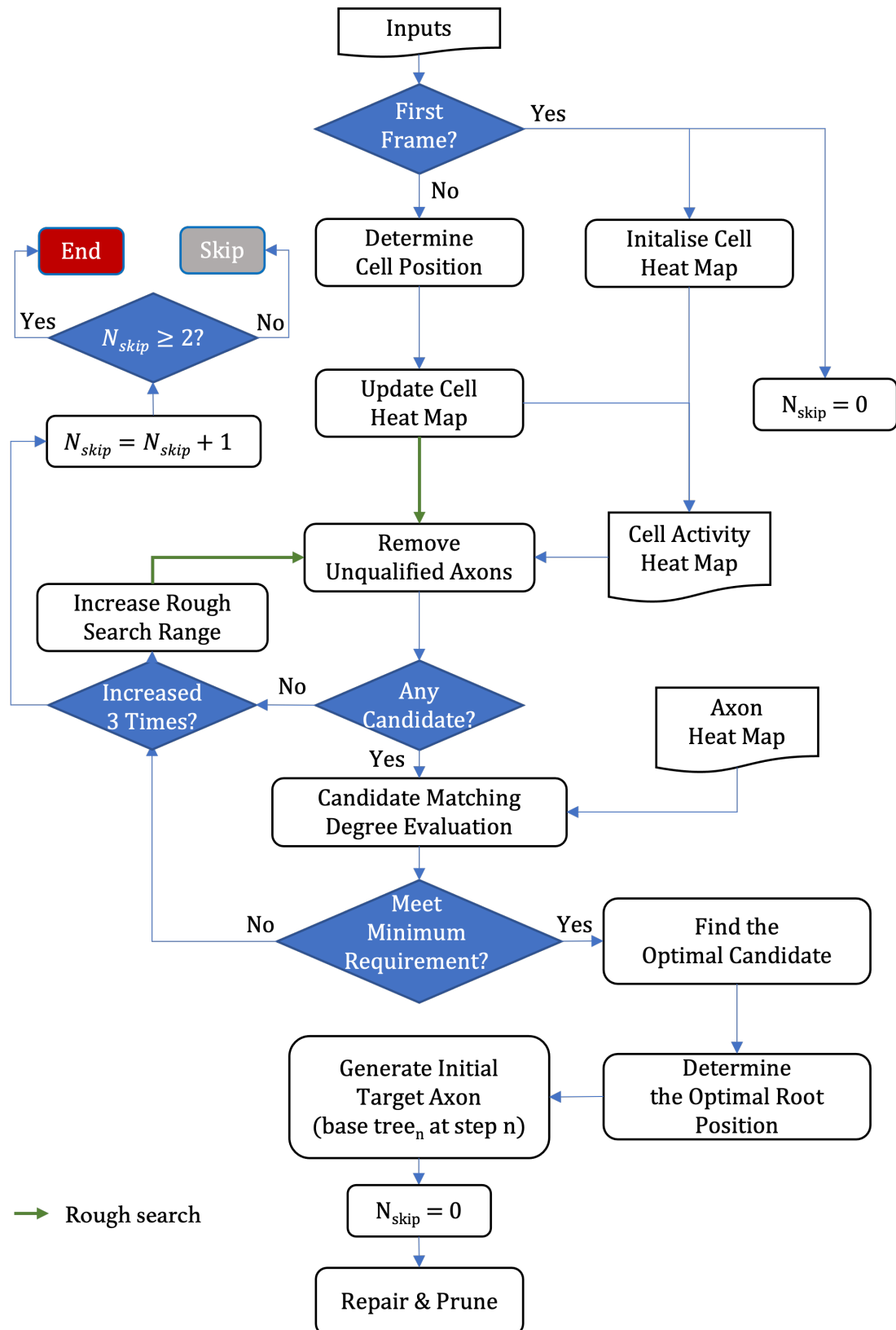


Figure 5.15: The scheme of creating the initial main tree in a frame, which is a detailed plot of the step 'Generate Initial Target Axon' in Fig. 5.12. The input of the module is the secondary axon profiles, and the output will be fixed and refined by the three-round repair and prune functions.

3. Find the axons in the list that touched the parent cell. If no axon meets this requirement, increase the cell area by  $20 * i$  pixels, keep searching. Increase the value of  $i$  ( $i \in \{1, 2, 3\}$ ), until first listed axon detected.
4. If still no axon is found, the system will skip this frame and repeat the last record as a new attachment to the sequence. Skipping three frames continuously will end the tracking process.

If only one axon candidate is detected in a run, output the axon. Otherwise, calculate the matching degree of every axon using the equation.

$$S(axon) = \frac{(S_{stem} + S_{major})}{2} \quad (5.15)$$

5. The minimum requirement of an initial axon trace is its  $S_{major} > 0.05$ . The one with the highest score will be selected as the target axon. Retrieve its potential root position from the primary profile. If there are several potential roots, determine the optimal root by the one with minimum  $U(x, y)$ , where

$$U(x, y) = D((x_c, y_c), (x, y)) + D((x', y'), (x, y)) \quad (5.16)$$

Here  $(x_c, y_c)$  are the co-ordinates of cell centre, and  $(x', y')$  are the previous root co-ordinates. Function  $D((x_1, y_1), (x_2, y_2))$  calculates the Euclidean distance be

Once the target axon in the frame is found. Construct the main tree with the information of the target axon in the primary profile.

### 5.3.6 Stop Tracking Conditions

There three conditions that the system would end the tracking process,

1. No matched initial axon was found for 2 continuous frames.
2. Have reached the end of a video.
3. Have reached the stop frame set by user.



---

**Algorithm 5** Find Target Axon

---

```

1: function FINDTARGETAXON(heatMap, heatMapCell, axons, cells, profiles,
   skippedNum)
2:   candiList = roughSearch(profiles)
3:   parentCell = heatMapCell(cells)
4:   for i in range(3) do
5:     temp = []
6:     scores = []
7:     for candi in candiList do
8:       condition1 = overlap(candi, parentCell.(dilation=i*20))
9:       condition2 = getMajorP(candi, heatMap)
10:      if condition1 and condition2 > 0.05 then
11:        temp.append(candi)
12:        s = (condition2+profiles.getStemP(candi, heatMap))/2
13:        scores.append(s)
14:      if len(temp) != 0 then
15:        break
16:    refinedList = temp
17:    if len(refinedList) == 0 then
18:      skippedNum += 1
19:      if skippedNum == 2 then
20:        return "end tracking"
21:    else
22:      skippedNum = 0
23:      target = refinedList[getMaxIndex(scores)]
24:    return target, skippedNum

```

---

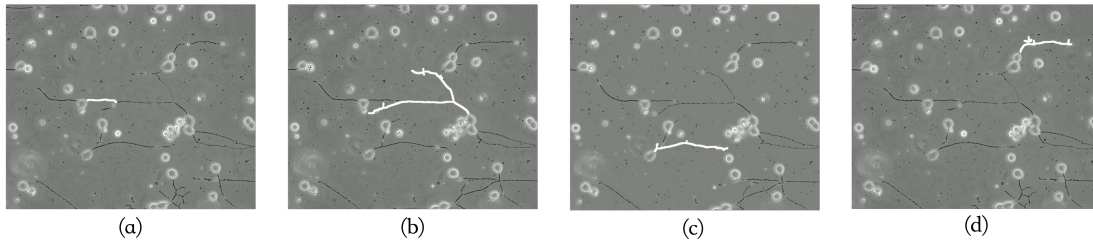


Figure 5.16: The results of tracking axons. Case (a)- (d) were collected from the same microscopy video. The target object in case (a) is encountering another axon sample. Case (b) and (c) show that the model can manage the situation where target axon is crossed by other samples from different directions. Case (d) demonstrates that Model-1 can repair a target trace by choosing the right candidate pieces and reconnecting them to appropriate positions.

## 5.4 Conclusion

In this chapter, we introduced Model-1 as a preliminary plan for tracking axons throughout videos. It can produce surprisingly accurate results which were used in the analysis in Chapter 6. The main challenge it addresses was axon entanglement and fracture fixation. Both issues can be detected and well managed by the two-stage checking and iterative cut and connect process. Besides, Fig. 5.16 demonstrated the case that model-1 is aware of the relationships between axons and cells even for the emerging tips of a target axon after crossing another existing trace.

The model still has major weaknesses which Model-2 will address. First, the current scheme takes a one-case-one-time criteria. It is convenient for researchers to sift valuable samples when adding necessary manipulation. The operational complexity increases. Model-1 cannot get rid of human correction, as it is tuned to the condition for the most common cases. The robustness of Model-1 is limited by the knowledge of the dataset, which means problems could occur in unseen cases.

As a solution to these challenges, we designed the graph-based tracking Model-2.

# Chapter 6

## Experiment Analysis and Model-2

The AxonTree structures the graphic data in an appropriate form for modelling axon activities. This enables statistic tools and graph algorithms to be more easily applied. Such a tree can be studied as a standardised object. Functions include visualisation (t-SNE, U-map), variable decomposition (PCA), graph understanding theories, as well as many deep-learning-based clustering and prediction algorithms are then feasible in this problem.

Essential information has been collected in the first phase of tracking. These data have two uses, for strengthening tracking quality and studying axon behaviours. We demonstrate two applications in this chapter, the model-2 for tracking axons and an algorithm developed for analysing experiments designed to study how neurons react to chemicals when growing.

### 6.1 Drug Experiment Analysis

In the drug tests, axon samples are considered together with the biomedical cues. The goal is to learn axon behaviour, including their response type and the response magnitude in different conditions. Our system can provide a systematic sampling plan with standardised measurement, which is more efficient and accurate than the current manual approaches.

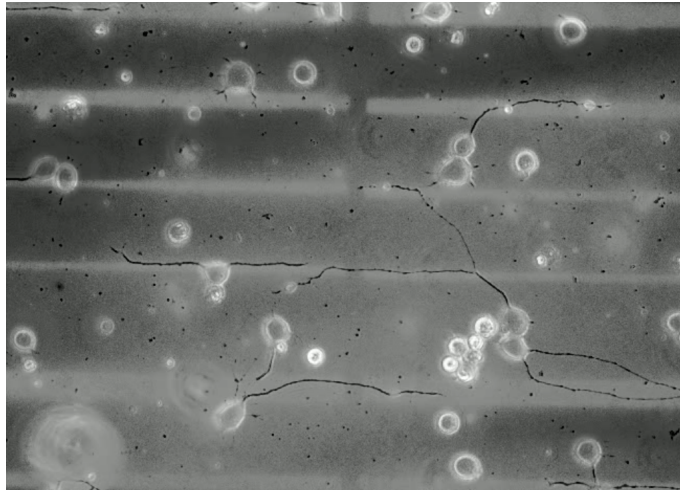


Figure 6.1: A demonstration of the concepts in this chapter. The figure is a synthesised picture of an axon activity image and a cue pattern image, which are the bright strips in the background. In this frame, the axons are attracted by the cues and tend to grow along the strips.

### 6.1.1 Experiment Introduction

The experiment measures the response of axons when they encounter various chemicals in the environment. An axon's growth is led by a support structure called growth-cone (the end of an axon as shown in Fig.6.1). It is a key object in this experiment. In the analysis, a response is represented by an axon's length extension velocity (growth speed  $\mu\text{m/s}$ ).

Multiple chemicals in multiple forms are expected to be tested. These are put on a plate as parallel strips with different widths. An example is shown in Fig.6.2.

The input has two parts, a video of axon activity and a picture in which the chemical strips are visible, both recorded by microscope. The chemical patterns are not visible under the same camera for recording axons and neural cells. Thus they are stored separately. The strip location is fixed, so only one picture is needed for each video.

The goal is to measure the speed of the growth-cones, their locations relative to the nearest strip and the type of the nearest strip.

### 6.1.2 Strip Recognition

The image quality of drug strips varies from blurry to clear. A challenge is separating these strips from the background properly. There can be many unexpected conditions.

We designed a simpler method to solve this problem. It builds strip masks based

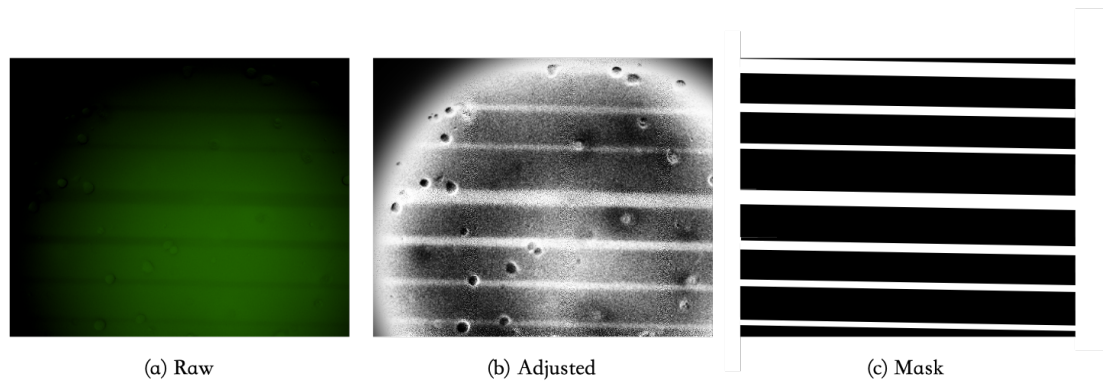


Figure 6.2: The demonstration of strip extraction procedures. The raw images were first be converted to a pre-defined uniform condition by adjusting their intensity histograms. The adjusted images were be segmented by a Canny edge filter to draw a crude mask. Since the signals in this edge mask can be skewed in many conditions, a linear function is used to estimate a collective decision based on the consensus of both sides of a strip. A refined mask is created based on this linear function.

on the consensus of the preliminary edge detection results. An example is shown in Figure.6.2

This method has four steps, as,

### Step 1: Image Adjusting

The images of the strips varied considerably. We performed an intensity normalisation based on finding the peaks (corresponding to strips and background) in the intensity histogram.

### Step 2: Edge Detection

The experiment requires a precise and straight strip boundary to determine whether a growth-cone has left the strip. The normalised image was be first processed by a Canny filter. Given the premise of only having (almost) horizontal strips, a coarse binary mask is drawn with a threshold. This threshold is estimated from the averaged x-axis in which strip parts are significantly brighter. Pixels belonging to the same side of an edge were connected by horizontal searching. The detected edges were sorted into pairs by their vertical order. The vertical pixels coordinates of one edge were rectified to remove obvious outliers.

### Step 3: Edge Regression

We fitted a line of the form  $y = ax + b$  to the edge pixels on each side of each strip. There could be an issue of having the two edge functions of a strip not

parallel after training. So the gradients were further calibrated by averaging all the strips in an image.

#### Step 4: Mask Generation

A refined binary mask was created based on these linear functions, in which the gap between a pair of edges is marked as strip area.

### 6.1.3 Video Stabilisation

It was observed that camera position may shift during the time of a video. This will impact the quality, because the strip masks are static. In order to fix this issue, we adjusted video images by comparing their corner patches and cropped the images accordingly to correct for the shift.

### 6.1.4 Interaction Definition

A growth-cone covers a small region which we approximate with a disk of a given radius. For most wide strips, such a radius is insignificant, as an axon's on-strip and off-strip status can be easily separated. However, if a strip is as narrow as  $2\mu\text{m}$  or  $5\mu\text{m}$ , the influence of growth-cone radius is non-trivial.

The system allows us to make a precise estimate of the relative position of a growth-cone about a strip pattern. We put them into four classes, depending on the centre location, which are on-strip, leaving(inside), leaving(outside), and off-strip. In Fig. 6.3, if an axon's entire growth cone is inside the boundary, then it has an on-strip status. If only its centre locates inside but has part outside then it is a leaving(inside). When its centre also moves out of the boundary but not the entire growth-cone, it is a leaving(outside) case. The situation of a growth-cone completely outside a strip is called off-strip.

### 6.1.5 Measurement

#### 6.1.5.1 Growth Velocity

An axon's growth velocity ( $v_t$ ) is calculated by its length ( $l_t$ ) changes between frame  $t - 1$  and  $t$ ,

$$v_t = l_t - l_{t-1} \quad (6.1)$$

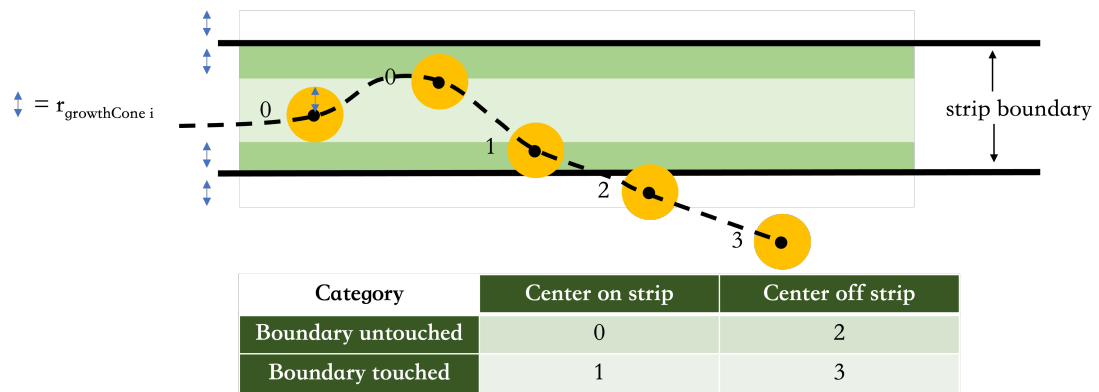


Figure 6.3: The illustration above shows how different interaction statuses are defined. There are four kinds, 1)on-strip, 2)leaving(inside), 3)leaving(outside), 4)off-strip. The class is determined by whether a growth-cone has its centre inside the boundary and touches the boundary. The detailed rules can be found in the table.

To be more specific, the length is measured as the Euclidean distance of the shortest line connecting all the pixels from an axon's growth-cone to its root, instead of the total number of pixels along a trace.

In the previous manual analysis, a growth velocity was measured as the moving distance of a tip. This can lead to inaccurate measurement when the camera is jittering, an axon may not grow but the position is changed. Or a cell, so stretching the connected axons. We chose the new measurement as these scenarios occur often in the data. One can find some minor differences between the figures produced by our system and manual analysis due to this modification. We believe that this will improve the precision and not influence the general conclusions.

### 6.1.6 Results

We analysed 78 videos from 5 individual experiments with our system. The videos had 97 frames each. A number of 207 axons are collected in total, including 22 samples about 2-micrometer strips, 44 samples about 5-micrometer strips, 40 samples about 10-micrometer strips, 33 samples about 20-micrometers strips, and the rest are the no-interaction cases.

Looking into the growth data presented in Fig. 6.4, the axons that interacted with the same type of strips are compared.

An axon may not always stay with one strip. It can leave a strip and switch to another. Fig. 6.5 shows the growth activity with status highlighted. Two examples

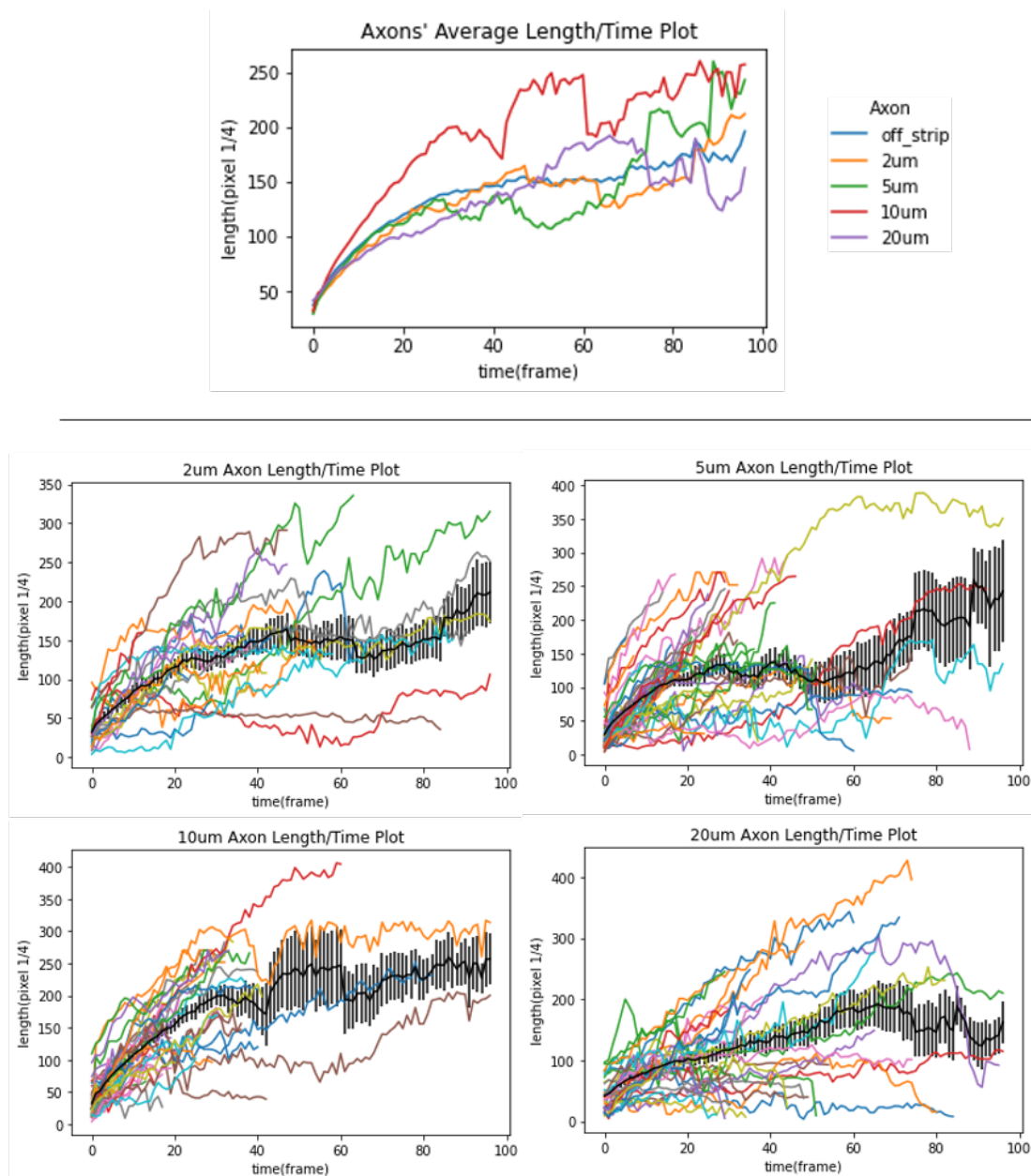


Figure 6.4: These plots show how axons interact with the substance strips in the widths from 2 micrometers to 20 micrometers. The black lines indicate the averaged lengths with error bars (SEM). These averaged lines are shown together with colours in the top plot. It is found the 10 micrometer strip has a clear advantage in boosting axon growth at the early stage.



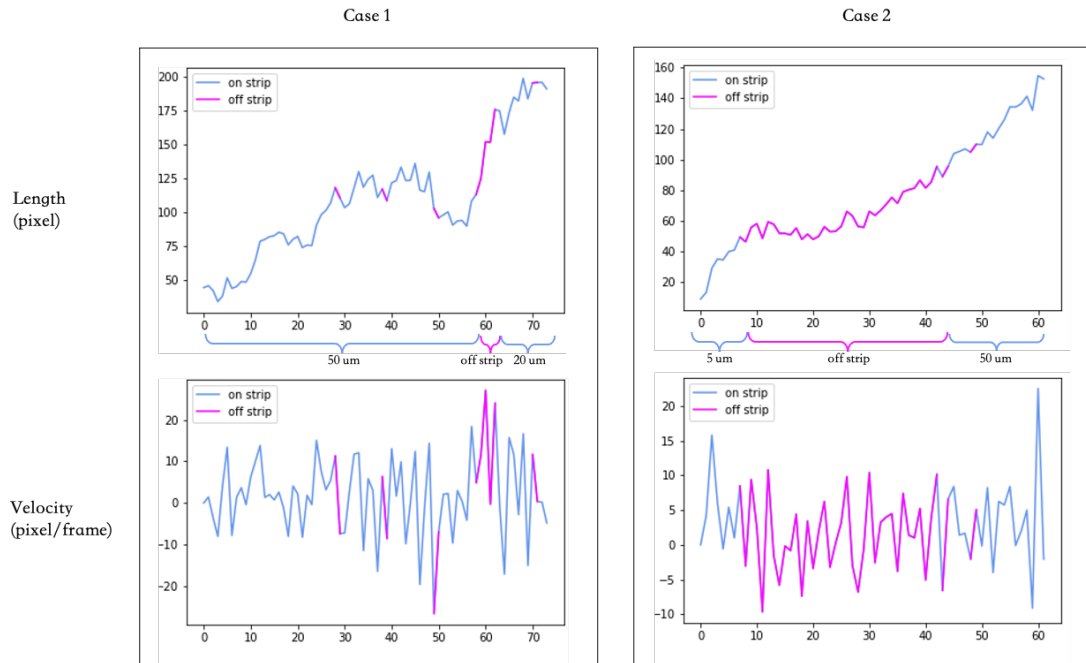


Figure 6.5: Cases with axon object transiting multiple strips. Two charts are presented for each case, a length chart and a velocity chart. Pink and blue colours are used to indicate the status of an axon, either on strip or off-strip. The brackets indicate the strip width that the axon is interacting with.

are demonstrated. The switching behaviour can be studied on frame basis. In manual analysis, information is mostly analysed on sample basis.

Since we had the location of the entire trace, we could also study the velocity on specific cases, shown as Fig. 6.6. In the first chart, the axon growth records on particular types of strips are illustrated. It should be noticed that the velocity distribution of the 10μm case is more skewed to the positive, showing on the histogram as a significantly higher mean growth velocity. And the 2μm distribution is more concentrated around zero, which renders a lower mean compared to the other cases.

The second chart shows the distribution of velocities for axons on the margin of each strip, defined as when part of the growth cone disk touches the edge of the strip. It demonstrates that axons at the margin behave differently from those away from the margin - this is particularly clear in the 2μm case. The 2μm strip is the most narrow pattern in this experiment, which has a much higher margin record ratio. The 20μm case, however, was not observed having frequent direction changes on its boundary.

Besides the normal statistics, the system can also look into the activity pattern with the techniques such as t-SNE and PCA, illustrated in Fig. 6.7. Comparing the

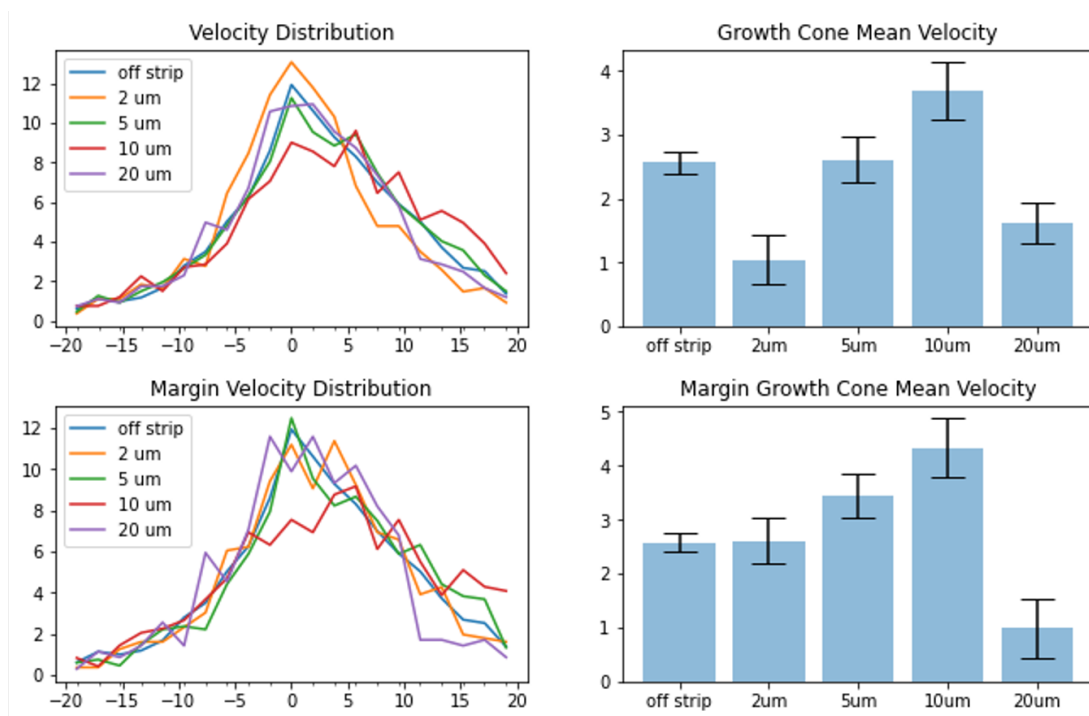


Figure 6.6: Axon growth pattern studied by the relative position of growth-cones. The term 'margin' in the first plot refers to the cases of a growth-cone touching any strip boundary.

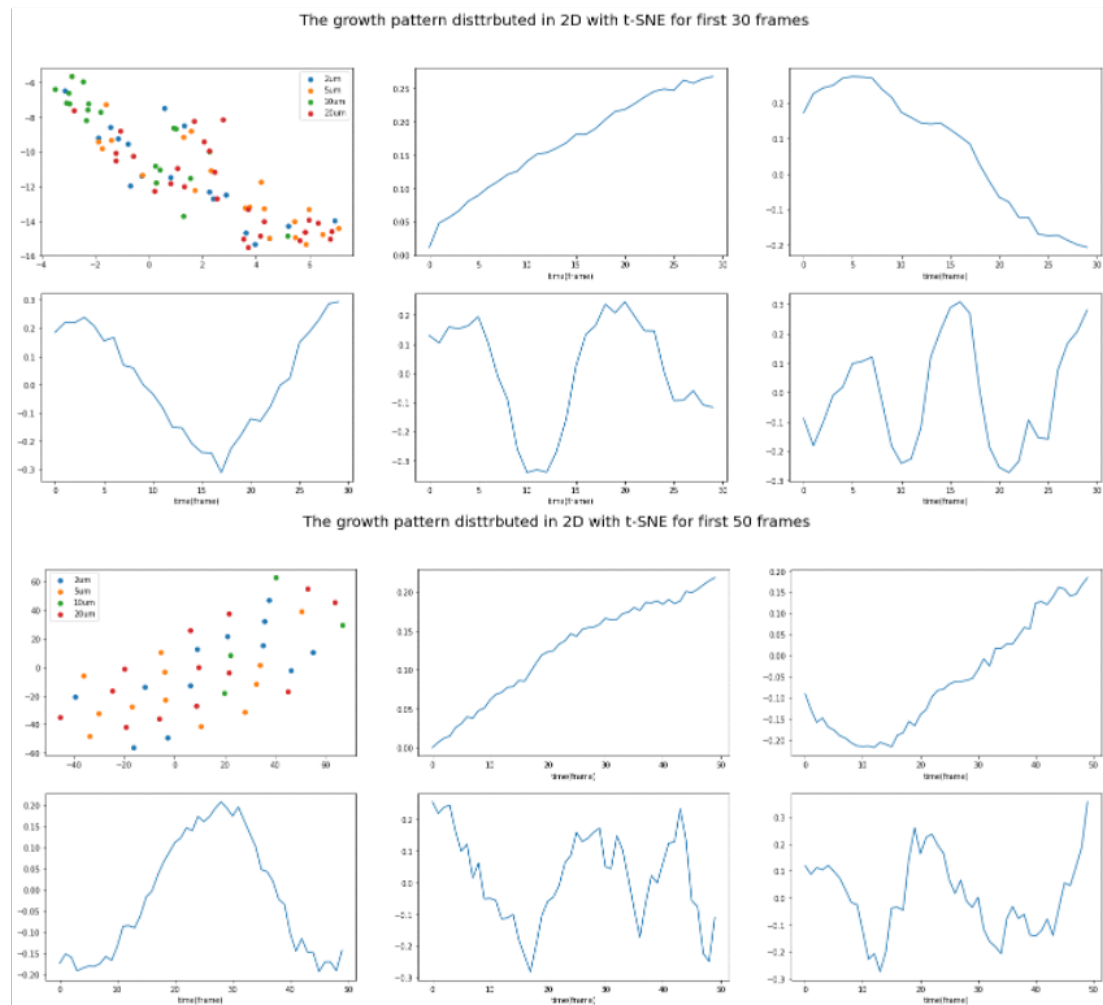


Figure 6.7: The t-SNE scatter plots and the principle components of the axons' first-30-frame and first-50-frame length records.

length growth pattern of axons in the first-30-frame to the first-50-frame, the two scales have similar primary growth patterns (primary principle component). Their secondary components are different, where the first-30-frame one shows a slope and the first-50-frame plots a stable increase after a shallow dip.

In the first-30-frame t-SNE scatter chart, the green dots which represent the 10um samples are clustered at one corner, while the other cases are clustered at the opposite. This confirms the average velocity chart observation in which the velocity on the 10um strips is significantly higher than the other categories.

### 6.1.7 Discussion

The automated system can analyse much larger samples than can be annotated by a human, enabling more robust results and conclusions to be drawn. The algorithm recycles the samples that were unworthy to measure due to limited value (too short in time, too small, etc.). These samples are cleaned with a standard process.

This section demonstrated an example of how the system can work in a drug study task. In the next section, we will introduce a slightly more complex task, where the analysis is based on higher dimensional features and requires an implicit induction. We make use of a graph neural model to make the decision.

## 6.2 Model 2: AxonTree Model

In Chapter 5, we introduced the tracking Model-1. AxonTrees can be analysed by graph methods. A good application example of this type is Model-2, in which the original axon tracking problem is turned into a graph separation task using the labels and features collected by Model-1. The new model uses the method of Graph Convolution(GCN)

### 6.2.1 Background

The model requires human correction because its iterative cutting and connecting process is developed on the knowledge about the most common cases. It has a relatively simple filtering mechanism and is conducted on the critical nodes of axon trees. In order to improve the performance, we designed Model-2 which is intrinsically a classifier of these critical nodes.

To extract complete axon topology, the classification is operating on an over-connected graph. This means the initial main tree will be connected with every nearby piece for ensuring all real nodes are included before pruning.

With the help of Model-2, the key functions in this tracking algorithm is simplified. Its iterative process is substituted by one connecting and one pruning (Model-2) operation. The tracking accuracy is increased with less graph computation.

### 6.2.2 Dataset Construction

The detailed procedure of annotation has been introduced in Chapter 5.

The input of an AxonTree model has two parts, a feature matrix, and an edge matrix. The features are collected using model-1 but stopped it before the iterative pruning and repairing. Thus this axon tree would keep every node that is found some degree of relevancy to the object.

The samples were from 19 videos with a range of 30 to 96 frames. One axon sample is taken from each frame. A total number of 1033 samples were randomly separated into two sets, 933 for training and 100 for testing. A learning rate of 0.001 is used for training Model-2 and initialised randomly.

### 6.2.3 Feature Matrix

In order to prepare the tree features, the system would first determine the best root node and all the connected nodes of it as the initial tree. Then, a rough searching (Chapter 5 Section.5.3.2) is conducted to find and merge (Chapter 5 Section.5.3.4) all nearby nodes. The process ensures an axon tree object has all potential nodes attached. And it will be turned into a feature matrix. Each matrix represents an instance in a frame with at least two rows. And each row represents a node of an AxonTree.

The version of model-2 described in this section has 12 input channels. They are,

**Attr. 1 & 2:** The x and y coordinates of a node.

**Attr. 3:** The length of the attached edge.

**Attr. 4:** The averaged value of the attached edge on the segmentation map.

**Attr. 5 - 7:** The embedding of the attached edge over the pixel features.

**Attr. 8:** The mean direction of the attached edge from the orientational index map calculated by the method described in Section.6.2.3.1.

**Attr. 9:** The trace distance from this node to the root.

**Attr. 10:** The score  $S_{\text{stem}}$  defined in Eq.5.11 Chapter 5.

**Attr. 11:** The score  $P_{\text{history}}$  defined in Eq.5.3 Chapter 5.

**Attr. 12:** The number of furcation times before this node.

#### 6.2.3.1 Majority Direction Voting

The direction labels along an edge in orientational index map would encounter a crossing border problem when estimating the direction expectation. Because they are circular labels. In the 16-directional (Symmetric GMASC) case, the axon pixel labelled as 15 ( $\theta = \frac{15}{8}\pi$ ) has a very close orientation with a nearby pixel labelled as 1 ( $\theta = \frac{1}{8}\pi$ ), the mean of them is  $8(\theta = \pi)$  rather than  $0(\theta = 0)$  which is the proper answer.

We use a majority voting strategy to address this issue. A number of 16 bins are created representing 16 directions of an axon edge. Every time getting a new pixel label, the system will case one vote to the bin of the label and 0.7 votes to consecutive bins respectively. We assume the standard deviation of a direction label at a position is 1, then 0.7 is selected according to the standard normal distribution table to include

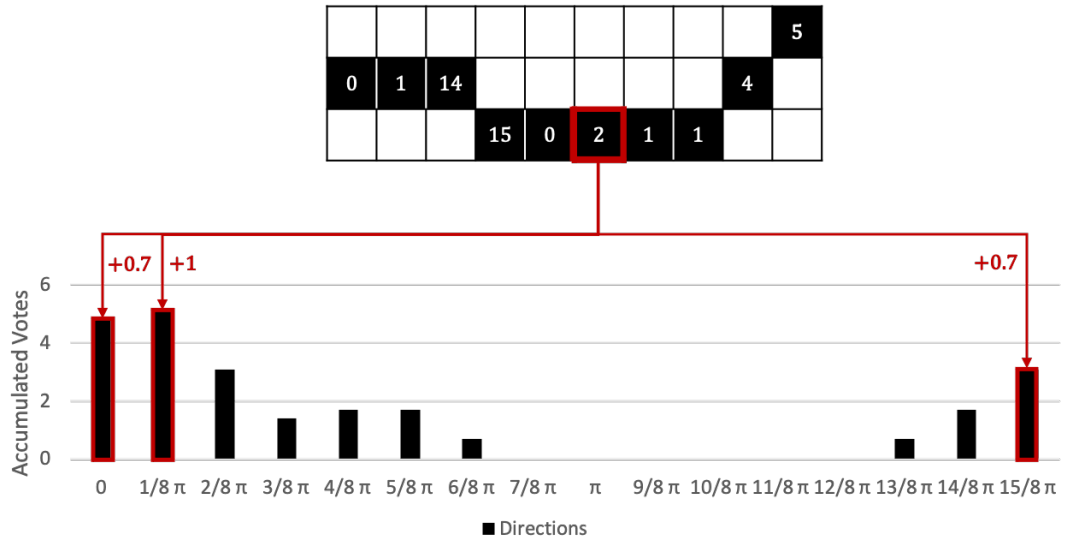


Figure 6.8: Axon edge direction expectation measurement  
The process of calculating direction expectation of an axon edge.

sufficient confidence. The bin with the maximum votes is selected as the direction expectation of the edge.

The process is illustrated in Fig.6.8. The outputs will be treated as a feature for model-2 in the node classification task.

## 6.2.4 Edge Direction Matrix

An edge direction matrix is the adjacent matrix of a graph in this context. It defines the direction of message integration, where only the information from a node with edge defined will be received. It is like a gate with the unconnected nodes given zero attention weights. For example, in the simplest case, a two-node graph, and the edge directs from node-1 to node-2. Then the edge direction matrix would look like  $\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$ , in which only the in-side node gets an open gate.

## 6.2.5 Methodolgy

We are using a model as illustrated in Fig. 6.9. The structure has four graph convolutional layers [61] and has batch normalisation and ReLU activate the function in between. Since this is a classification task, the output would be a single label for each node.

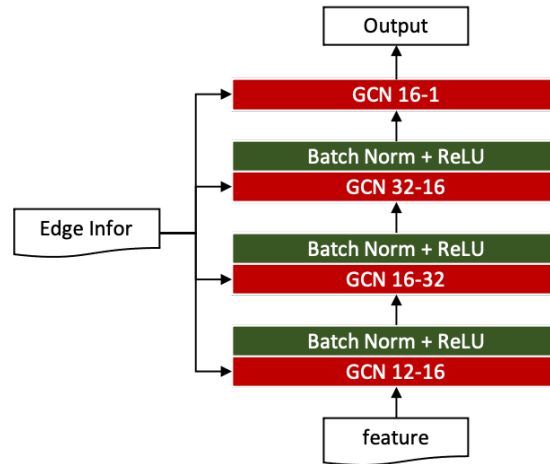


Figure 6.9: The graph neural network structure for classifying axon nodes to achieve an instance separation effect. Because axon trace entanglement is hard to be solved based on a sole image. The graph model receives encoded messages as a part of features from the previous frame to make a joint decision on nodes.

Table 6.1: A comparison between model-1 and model-2 on the axon tree dataset

Weights	Acc	AUC
Model 1	83.8	-
Model 2	96.6	99.4

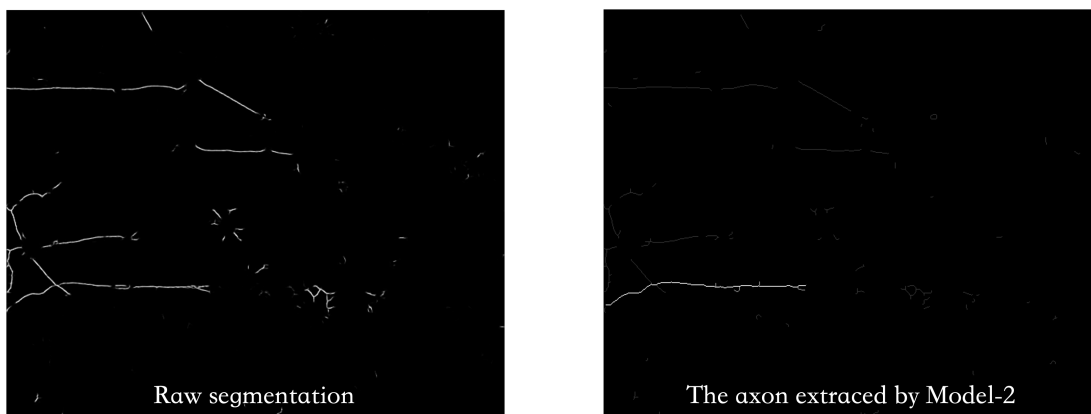


Figure 6.10: Model-2 is also able to detach traversing axons and reconnect broken traces. The left plot is a binary segmentation map, the right plot shows the corresponding tracking output in which the left bottom axon is selected as the target object.



### 6.2.6 Performance Comparison

The dataset is constructed with boundary cases. Compared to the output of model-1, model-2 has significantly better accuracy, as shown in Table.6.1. Like Model-1, the graph based Model-1 is also able to detach entangled axons as the example illustrated in Fig. 6.10.

The two models take a slightly different processing order as Model-2 only has one round of reconnection. This may mislead the system by linking a candidate piece to an irrelevant piece which is initially connected to the main tree due to its closer position. This issue is rare to see in practice, because the piece between two related pieces are more likely to be relevant as well.

Compared with Model-1, Model-2 makes decision for all nodes. Even a middle node has a negative flag, the nodes after it will be checked. In Model-1, if a selected linking position is not the correct choice, the rest nodes on this piece will not be re-connected to the main tree. This protect an axon trees in Model-2 from consequential damages.

### 6.2.7 Conclusion

In this chapter, we introduced a node classifier, Model-2. Its inputs include node features and edge maps on over-connected axon graphs. Compared to Model-1, it is more efficient as it reduces the three rounds connect and cut to a single round. In the experiment, we also found that Model-2 can generate much more complete axon objects.

# Chapter 7

## Conclusion

We have introduced a system and the related algorithms for studying neuron cell and axon activity in microscopy videos.

The system can be separated into three modules, a segmentation model for finding axon and cell objects, a two-step tracking scheme that can disentangle axons, and an analysis module to help generate experiment reports.

### 7.1 Contributions

The main lessons from the work in this thesis are as follows.

The multi-angle and scale (MASC) is a powerful network structure for segmenting objects in images. It requires far fewer parameters than comparable approaches such as U-Net. Because of this it requires fewer training samples to achieve good results.

The original MASC algorithm was developed for detecting curvilinear objects. The invariant-feature idea behind it has great potential, though we identified some issues related to the number of constraints on each filter. We designed an extension, general multi-angle and scale convolution (G-MASC), which dealt with these issues by introducing response shaping and showing how we could extract useful information from pattern property maps. Reference nodes were created to provide additional regularisation on the directional weights, and the orientation and scale property labels were embedded in latent features as extra information sources. The differences between MASC and G-MASC were evaluated with a series of experiments. We also identified which approach would be most suitable for different situations.

A history-adaptive tracking Model-1 and a training-based tracking Model-2 were proposed to find axons in videos. The Model-1 system can convert graphical axons

to an axonTree format, during which the key information was extracted. The graph neural network based Model-2 improved the quality of the original annotation, and was the first project using tree theory to solve 2D axon tasks. The results generated by the models were used for analysing videos from experiments studying the growth of axons and how they interact with chemicals on the substrate.

## 7.2 Future Works

The work could be taken further in several ways, including:

- 1) Integrating the property aware mechanism into the general neural networks.

The property maps are derived from equivalent features, which can be very useful in solving some tasks. For example, the scale information for a self-driving car can help control the distance and is faster than most post-processing methods.

- 2) Further optimising the MASC computation.

One possible solution was provided in Chapter 3 suggesting the choice between a sequential and a parallel scheme.

A more simple method is in the scale search step. We found that, for the axon task, once a model converged, the main scale of the model is stabilised. Instead of choosing a scale channel by the maximum response, the search can be managed only on the found main scale. Only using the main scale will decrease computation but not impact the modelling, because the area not matching with the main scale will have a lower intensity compared to the maximum signal.

- 3) Developing a robust graph refinement mechanism.

The process of axon tracking can be split into series of frame by frame steps which is a process of tree refinement. The accuracy on single axon frames was able to reach 96.6% (accuracy) and 99.4% (AUC). However, it is still not accurate enough to be made fully automatic for two reasons.

- A mistake may happen on a node closer to the root of an axon.
- A mistake may happen on a frame at the beginning of a video.

To alleviate the amplification effect of these two conditions, a robust refinement mechanism is required. Some attempts such as the reverse proof which initialises the tracking model from the end of video for double-checking the decision have produced some encouraging results.

4) Designing a new application for AxonTrees.

A key novelty of the system is it represents a biological object by a tree structure. The graph tools can be used on these objects for extracting latent information.

# Bibliography

- [1] Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt, and Joan M Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.
- [2] Md Zahangir Alom, Mahmudul Hasan, Chris Yakopcic, Tarek M Taha, and Vijayan K Asari. Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation. 2018.
- [3] Thomas L Athey, Jacopo Teneggi, Joshua T Vogelstein, Daniel J Tward, Ulrich Mueller, and Michael I Miller. Fitting splines to axonal arbors quantifies relationship between branch order and geometry. *Frontiers in Neuroinformatics*, page 38, 2021.
- [4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.
- [5] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5221–5229, 2017.
- [6] Jeff K Beckett and John C Bancroft. Event detection in prestack migration using matched filters. *CREWES Research Report*, 14, 2002.
- [7] Nazanin Beheshti and Lennart Johnsson. Squeeze u-net: A memory and energy efficient image segmentation network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 364–365, 2020.
- [8] 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.
- [9] John Bowler, Rogerio Feris, Liangliang Cao, Jun Wang, and Mo Zhou. Automated axon segmentation from highly noisy microscopic videos. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 915–920. IEEE, 2015.
- [10] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. *arXiv preprint arXiv:1611.01576*, 2016.
- [11] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.
- [12] Jason Brownlee. A gentle introduction to the rectified linear unit (relu). *Machine learning mastery*, 6, 2019.
- [13] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.
- [14] Hongmin Cai, Xiaoyin Xu, Ju Lu, Jeff W Lichtman, SP Yung, and Stephen TC Wong. Repulsive force based snake model to segment and track neuronal axons in 3d microscopy image stacks. *NeuroImage*, 32(4):1608–1620, 2006.
- [15] Sijing Cai, Yunxian Tian, Harvey Lui, Haishan Zeng, Yi Wu, and Guannan Chen. Dense-unet: a novel multiphoton in vivo cellular image segmentation model based on a convolutional neural network. *Quantitative imaging in medicine and surgery*, 10(6):1275, 2020.
- [16] Joao Carreira, Rui Caseiro, Jorge Batista, and Cristian Sminchisescu. Semantic segmentation with second-order pooling. In *European Conference on Computer Vision*, pages 430–443. Springer, 2012.
- [17] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

- [18] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.
- [19] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. Tensormask: A foundation for dense object segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2061–2069, 2019.
- [20] Xiuyuan Cheng, Qiang Qiu, Robert Calderbank, and Guillermo Sapiro. Rotdcf: Decomposition of convolutional filters for rotation-equivariant deep networks. *arXiv preprint arXiv:1805.06846*, 2018.
- [21] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.
- [22] Yu Cheng, Felix X Yu, Rogerio S Feris, Sanjiv Kumar, Alok Choudhary, and Shi-Fu Chang. An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE international conference on computer vision*, pages 2857–2865, 2015.
- [23] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.
- [24] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.
- [25] Taco S Cohen and Max Welling. Steerable cnns. *arXiv preprint arXiv:1612.08498*, 2016.
- [26] Hanjun Dai, Zornitsa Kozareva, Bo Dai, Alex Smola, and Le Song. Learning steady-states of iterative algorithms over graphs. In *International conference on machine learning*, pages 1106–1114. PMLR, 2018.
- [27] Tianhong Dai, Magda Dubois, Kai Arulkumaran, Jonathan Campbell, Cher Bass, Benjamin Billot, Fatmatulzehra Uslu, Vincenzo De Paola, Claudia Clopath, and Anil Anthony Bharath. Deep reinforcement learning for sub-pixel neural tracking. In *International conference on medical imaging with deep learning*, pages 130–150. PMLR, 2019.

- [28] Neda Davoudi, Xosé Luís Deán-Ben, and Daniel Razansky. Deep learning optoacoustic tomography with sparse data. *Nature Machine Intelligence*, 1(10):453–460, 2019.
- [29] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv preprint arXiv:1606.09375*, 2016.
- [30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [31] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [32] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. *Advances in neural information processing systems*, 30, 2017.
- [33] 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.
- [34] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019.
- [35] Nicolás Gaggion, Lucas Mansilla, Diego Milone, and Enzo Ferrante. Hybrid graph convolutional neural networks for landmark-based anatomical segmentation. *arXiv preprint arXiv:2106.09832*, 2021.
- [36] Hongyang Gao and Shuiwang Ji. Efficient and invariant convolutional neural networks for dense prediction. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 871–876. IEEE, 2017.



- [37] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1416–1424, 2018.
- [38] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7036–7045, 2019.
- [39] Hossein Gholamalinezhad and Hossein Khosravi. Pooling methods in deep neural networks, a review. *arXiv preprint arXiv:2009.07485*, 2020.
- [40] Rohan Ghosh and Anupam K Gupta. Scale steerable filters for locally scale-invariant convolutional neural networks. *arXiv preprint arXiv:1906.03861*, 2019.
- [41] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
- [42] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216*, 2017.
- [43] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, pages 297–312. Springer, 2014.
- [44] Ali Hatamizadeh, Shilpa P Ananth, Xiaowei Ding, Demetri Terzopoulos, Nima Tajbakhsh, et al. Automatic segmentation of pulmonary lobes using a progressive dense v-network. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, pages 282–290. Springer, 2018.
- [45] Junjun He, Zhongying Deng, Lei Zhou, Yali Wang, and Yu Qiao. Adaptive pyramid context network for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7519–7528, 2019.
- [46] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of*

- the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.
- [47] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.
- [48] 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.
- [49] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.
- [50] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [51] Emiel Hoogeboom, Jorn WT Peters, Taco S Cohen, and Max Welling. Hexaconv. *arXiv preprint arXiv:1803.02108*, 2018.
- [52] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [53] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3464–3473, 2019.
- [54] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.
- [55] Jingfei Hu, Hua Wang, Shengbo Gao, Mingkun Bao, Tao Liu, Yaxing Wang, and Jicong Zhang. S-unet: A bridge-style u-net framework with a saliency mechanism for retinal vessel segmentation. *IEEE Access*, 7:174167–174177, 2019.
- [56] Lang Huang, Yuhui Yuan, Jianyuan Guo, Chao Zhang, Xilin Chen, and Jingdong Wang. Interlaced sparse self-attention for semantic segmentation. *arXiv preprint arXiv:1907.12273*, 2019.

- [57] Qin Huang, Chunyang Xia, Chihao Wu, Siyang Li, Ye Wang, Yuhang Song, and C-C Jay Kuo. Semantic segmentation with reverse attention. *arXiv preprint arXiv:1707.06426*, 2017.
- [58] Vassilis N Ioannidis, Antonio G Marques, and Georgios B Giannakis. Graph neural networks for predicting protein functions. In *2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 221–225. IEEE, 2019.
- [59] 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.
- [60] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.
- [61] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- [62] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9799–9808, 2020.
- [63] Marek Kowal, Michał Żejmo, Marcin Skobel, Józef Korbicz, and Roman Monczak. Cell nuclei segmentation in cytological images using convolutional neural network and seeded watershed algorithm. *Journal of digital imaging*, 33(1):231–242, 2020.
- [64] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.
- [65] Ashish Kumar, RK Agrawal, and Leve Joseph. Itermiunet: A lightweight architecture for automatic blood vessel segmentation. *arXiv preprint arXiv:2208.01485*, 2022.
- [66] Neeraj Kumar, Ruchika Verma, Sanuj Sharma, Surabhi Bhargava, Abhishek Vahadane, and Amit Sethi. A dataset and a technique for generalized nuclear segmentation for computational pathology. *IEEE transactions on medical imaging*, 36(7):1550–1560, 2017.

- [67] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.
- [68] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [69] Tao Lei, Yu Zhang, Sida I Wang, Hui Dai, and Yoav Artzi. Simple recurrent units for highly parallelizable recurrence. *arXiv preprint arXiv:1709.02755*, 2017.
- [70] Bo Li, Qili Wang, and Gim Hee Lee. Filtra: Rethinking steerable cnn by filter transform. *arXiv preprint arXiv:2105.11636*, 2021.
- [71] Hanchao Li, Pengfei Xiong, Jie An, and Lingxue Wang. Pyramid attention network for semantic segmentation. *arXiv preprint arXiv:1805.10180*, 2018.
- [72] Lu Li, Chao Wang, Hong Zhang, and Bo Zhang. Residual unet for urban building change detection with sentinel-1 sar data. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 1498–1501. IEEE, 2019.
- [73] Xia Li, Zhisheng Zhong, Jianlong Wu, Yibo Yang, Zhouchen Lin, and Hong Liu. Expectation-maximization attention networks for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9167–9176, 2019.
- [74] Xiaomeng Li, Hao Chen, Xiaojuan Qi, Qi Dou, Chi-Wing Fu, and Pheng-Ann Heng. H-denseunet: hybrid densely connected unet for liver and tumor segmentation from ct volumes. *IEEE transactions on medical imaging*, 37(12):2663–2674, 2018.
- [75] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.
- [76] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

- [77] Bo Liu, Lin Gu, and Feng Lu. Unsupervised ensemble strategy for retinal vessel segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 111–119. Springer, 2019.
- [78] Xinyu Liu and Xiaoguang Di. Tanhexp: A smooth activation function with high convergence speed for lightweight neural networks. *IET Computer Vision*, 15(2):136–150, 2021.
- [79] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.
- [80] Zewen Liu and Timothy Cootes. Masc-units: Training oriented filters for segmenting curvilinear structures. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 590–599. Springer, 2021.
- [81] Zewen Liu, Timothy Cootes, and Christoph Balleström. An end to end system for measuring axon growth. In *International Workshop on Machine Learning in Medical Imaging*, pages 455–464. Springer, 2020.
- [82] Zhenghao Liu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. Fine-grained fact verification with kernel graph attention network. *arXiv preprint arXiv:1910.09796*, 2019.
- [83] 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.
- [84] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6232–6240, 2017.
- [85] Shangzhen Luan, Chen Chen, Baochang Zhang, Jungong Han, and Jianzhuang Liu. Gabor convolutional networks. *IEEE Transactions on Image Processing*, 27(9):4357–4366, 2018.
- [86] Diego Marcos, Benjamin Kellenberger, Sylvain Lobry, and Devis Tuia. Scale equivariance in cnns with vector fields. *arXiv preprint arXiv:1807.11783*, 2018.

- [87] Alessio Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.
- [88] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. IEEE, 2016.
- [89] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5115–5124, 2017.
- [90] Agata Mosinska, Pablo Marquez-Neila, Mateusz Koziński, and Pascal Fua. Beyond the pixel-wise loss for topology-aware delineation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3136–3145, 2018.
- [91] Sajjad Mozaffari, Omar Y Al-Jarrah, Mehrdad Dianati, Paul Jennings, and Alexandros Mouzakitis. Deep learning-based vehicle behavior prediction for autonomous driving applications: A review. *IEEE Transactions on Intelligent Transportation Systems*, 23(1):33–47, 2020.
- [92] Jawad Nagi, Frederick Ducatelle, Gianni A Di Caro, Dan Cireşan, Ueli Meier, Alessandro Giusti, Farrukh Nagi, Jürgen Schmidhuber, and Luca Maria Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pages 342–347. IEEE, 2011.
- [93] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European conference on computer vision*, pages 483–499. Springer, 2016.
- [94] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.
- [95] José Ignacio Orlando, Elena Prokofyeva, and Matthew B Blaschko. A discriminatively trained fully connected conditional random field model for blood vessel

- segmentation in fundus images. *IEEE transactions on Biomedical Engineering*, 64(1):16–27, 2016.
- [96] Christopher G Owen, Alicja R Rudnicka, Robert Mullen, Sarah A Barman, Dorothy Monekosso, Peter H Whincup, Jeffrey Ng, and Carl Paterson. Measuring retinal vessel tortuosity in 10-year-old children: validation of the computer-assisted image analysis of the retina (caiar) program. *Investigative ophthalmology & visual science*, 50(5):2004–2010, 2009.
- [97] Ashutosh Pandey and DeLiang Wang. Densely connected neural network with dilated convolutions for real-time speech enhancement in the time domain. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6629–6633. IEEE, 2020.
- [98] placeholder. placeholder. pages 1–365, 2021.
- [99] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [100] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.
- [101] 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.
- [102] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.
- [103] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.
- [104] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

- [105] Nahian Siddique, Paheding Sidike, Colin Elkin, and Vijay Devabhaktuni. U-net and its variants for medical image segmentation: theory and applications. *arXiv preprint arXiv:2011.01118*, 2020.
- [106] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [107] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.
- [108] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.
- [109] Joes Staal, Michael D Abramoff, Meindert Niemeijer, Max A Viergever, and Bram Van Ginneken. Ridge-based vessel segmentation in color images of the retina, 2004.
- [110] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [111] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.
- [112] Matej Ulicny, Vladimir A Krylov, and Rozenn Dahyot. Harmonic networks for image classification. In *BMVC*, page 202, 2019.
- [113] Jeya Maria Jose Valanarasu, Poojan Oza, Ilker Hacihaliloglu, and Vishal M Patel. Medical transformer: Gated axial-attention for medical image segmentation. *arXiv preprint arXiv:2102.10662*, 2021.
- [114] Koen EA Van de Sande, Jasper RR Uijlings, Theo Gevers, and Arnold WM Smeulders. Segmentation as selective search for object recognition. In *2011 International Conference on Computer Vision*, pages 1879–1886. IEEE, 2011.



- [115] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.
- [116] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
- [117] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2017.
- [118] Kaiping Wang, Bo Zhan, Chen Zu, Xi Wu, Jiliu Zhou, Luping Zhou, and Yan Wang. Triple uncertainty guided mean teacher model for semi-supervised medical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 450–460. Springer, 2021.
- [119] Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell. Understanding convolution for semantic segmentation. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 1451–1460. IEEE, 2018.
- [120] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018.
- [121] Yan Wang, Xu Wei, Fengze Liu, Jieneng Chen, Yuyin Zhou, Wei Shen, Elliot K Fishman, and Alan L Yuille. Deep distance transform for tubular structure segmentation in ct scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3833–3842, 2020.
- [122] Maurice Weiler and Gabriele Cesa. General  $e(2)$ -equivariant steerable cnns. *arXiv preprint arXiv:1911.08251*, 2019.
- [123] Maurice Weiler, Mario Geiger, Max Welling, Wouter Boomsma, and Taco Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *arXiv preprint arXiv:1807.02547*, 2018.

- [124] 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.
- [125] 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.
- [126] Qing Wu, Yuwei Li, Lan Xu, Ruiming Feng, Hongjiang Wei, Qing Yang, Boliang Yu, Xiaozhao Liu, Jingyi Yu, and Yuyao Zhang. Irem: High-resolution magnetic resonance image reconstruction via implicit neural representation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 65–74. Springer, 2021.
- [127] Yicheng Wu, Yong Xia, Yang Song, Donghao Zhang, Dongnan Liu, Chaoyi Zhang, and Weidong Cai. Vessel-net: retinal vessel segmentation under multi-path supervision. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 264–272. Springer, 2019.
- [128] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.
- [129] Xiao Xiao, Shen Lian, Zhiming Luo, and Shaozi Li. Weighted res-unet for high-quality retina vessel segmentation. In *2018 9th international conference on information technology in medicine and education (ITME)*, pages 327–331. IEEE, 2018.
- [130] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.
- [131] Yutong Xie, Jianpeng Zhang, Chunhua Shen, and Yong Xia. Cotr: Efficiently bridging cnn and transformer for 3d medical image segmentation. *arXiv preprint arXiv:2103.03024*, 2021.

- [132] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057. PMLR, 2015.
- [133] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.
- [134] Rui Xu, Tiantian Liu, Xinchun Ye, Lin Lin, and Yen-Wei Chen. Boosting connectivity in retinal vessel segmentation via a recursive semantics-guided network. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 786–795. Springer, 2020.
- [135] Yichong Xu, Tianjun Xiao, Jiaying Zhang, Kuiyuan Yang, and Zheng Zhang. Scale-invariant convolutional neural networks. *arXiv preprint arXiv:1411.6369*, 2014.
- [136] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.
- [137] Liang Yuan and Junda Zhu. Video object tracking in neural axons with fluorescence microscopy images. *Journal of Applied Mathematics*, 2014, 2014.
- [138] Yuhui Yuan, Lang Huang, Jianyuan Guo, Chao Zhang, Xilin Chen, and Jingdong Wang. Ocnet: Object context network for scene parsing. *arXiv preprint arXiv:1809.00916*, 2018.
- [139] Tongjie Y Zhang and Ching Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984.
- [140] Yundong Zhang, Huiye Liu, and Qiang Hu. Transfuse: Fusing transformers and cnns for medical image segmentation. *arXiv preprint arXiv:2102.08005*, 2021.
- [141] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, 2018.
- [142] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

- [143] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. Psanet: Point-wise spatial attention network for scene parsing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 267–283, 2018.
- [144] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1529–1537, 2015.
- [145] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.
- [146] S Kevin Zhou, Hoang Ngan Le, Khoa Luu, Hien V Nguyen, and Nicholas Ayache. Deep reinforcement learning in medical imaging: A literature review. *Medical image analysis*, 73:102193, 2021.
- [147] Junda Zhu and Liang Yuan. Neurofilament tracking by detection in fluorescence microscopy images. In *2013 IEEE International Conference on Image Processing*, pages 3123–3127. IEEE, 2013.
- [148] Juntang Zhuang. Laddernet: Multi-path networks based on u-net for medical image segmentation. *arXiv preprint arXiv:1810.07810*, 2018.
- [149] Rodger E Ziemer and William H Tranter. *Principles of communications*. John Wiley & Sons, 2014.
- [150] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8697–8710, 2018.